Michel Archambault

MIEUX PROGRAMMER SUR AMSTRAD

SORACOM
informatique

MIEUX PROGRAMMER SUR AMSTRAD

« La loi du 11 mars 1957 n'autorisant, aux termes des alinéas 2 et 3 de l'article 41, d'une part que « les copies ou reproductions strictement réservées à l'usage privé du copiste et non destinées à une utilisation collective » et, d'autre part, que les analyses et les courtes citations dans un but d'exemple et d'illustration, « toute représentation ou reproduction intégrale, ou partielle, faite sans le consentement de l'auteur ou de ses ayants-droit ou ayants cause, est illicite » (alinéa premier de l'article 40). Cette représentation ou reproduction, par quelque procédé que ce soit, constituerait donc une contrefaçon sanctionnée par les articles 425 et suivants du Code Pénal. »

© Editions SORACOM — 1985

ISBN 2-904032-35-5

Michel Archambault

MIEUX PROGRAMMER SUR AMSTRAD

SORACOM editions

— Le Grand Logis —
10, avenue du Général de Gaulle
35170 BRUZ.

AVERTISSEMENT

Cet ouvrage est le complément pratique du manuel d'origine, il s'adresse donc à ceux qui connaissent au moins la majeure partie du Basic de l'AMSTRAD. Trois objectifs sont intimement mêlés :

• L'art de concevoir et de créer un programme d'une manière EFFICACE, c'est-à-dire rapidité, performances et fiabilité. Une approche donc de la qualité professionnelle ; nous en profiterons pour vous familiariser un peu avec le jargon de ce milieu très ouvert.

• Vous livrer une multitude d'astuces de programmation et de courts passages très utiles, que vous pourrez transférer et adapter dans vos œuvres.

• Expliquer clairement certains points très obscurs du manuel d'origine, en insistant aussi sur certaines fonctions d'aspect anodin dont vous ne soupçonnez peut-être pas l'utilité pratique.

Voilà pour le fond, voyons pour la forme :

La micro-informatique est pour vous un LOISIR, un jeu intellectuel ; donc notre souci permanent a été la CLARTE. Pas de cours magistraux à avaler du début jusqu'à la fin, mais des chapitres assez indépendants. Cela explique les quelques répétitions, car nous savons que vous ne lirez pas ce livre comme un roman... Si un chapitre vous semble (pour le moment) d'une compréhension difficile, ne vous obstinez pas et passez à un autre.

Tous les listings sont représentés en quarante caractères par ligne afin qu'ils représentent exactement ce que vous aurez à l'écran en faisant

LIST. Pas de longs programmes qui ne servent à rien, mais de nombreux ''passages'' courts très utiles ; parfois l'utilité est remplacée par le côté spectaculaire afin de montrer les ''dessous'' de certaines fonctions.

Deux remarques pour conclure :

• Dans la table des matières, pas un mot sur le langage machine, et ce pour deux raisons : un livre d'épaisseur double y suffirait à peine. Le Basic de l'AMSTRAD étant très complet et surtout très rapide, le recours à cette technique est beaucoup plus rare qu'avec la plupart des autres micro-ordinateurs.

• Tous les micro-ordinateurs sont différents, chacun a sa personnalité, ses avantages, ses inconvénients. L'art de la programmation consiste à s'appuyer sur les points forts afin de contourner les points faibles. De ce fait, la majeure partie de cet ouvrage est spécifique, adaptée à l'AMSTRAD CPC et à aucune autre machine.

Etes-vous prêt à décupler votre capacité de programmation ?

CHAPITRE I

LES DEUX URGENCES

Il s'agit de deux petits travaux fort simples, mais définitifs, qui vont accélérer considérablement la frappe d'un programme au clavier.

Le premier outil est un court programme sur cassette, *à exécuter à chaque mise sous tension,* qui va vous définir seize ''touches pour fainéants'' judicieusement choisies. Le second est une réglette en bristol (ou carton blanc) dont les graduations nous éviteront toutes les estimations de tabulations (TAB et LOCATE) pour centrer une chaîne.

LE PROGRAMME ''KEY''

Le chargement de ce programme ne demande que quatorze secondes.

Voyons d'abord ce que deviennent les touches de fonctions du pavé numérique :

''7'' = CLS + enter
''8'' = FOR I = 1 TO (+ 1 espace)
''9'' = EDIT (+1 espace)
''4'' = caractères noirs, PAPER et BORDER gris + enter
''5'' = caractères roses, PAPER bleu + enter
''6'' = KEY 134. C'est une touche libre ; il suffit d'ajouter la virgule, double cote (guillemets) et la fonction exemple :

,''RUN'' + CHR$(13)

"1" = LOCATE (+ 1 espace)
"2" = CHR$(
"3" = LIST (+ 1 espace)
"0" = PRINT (+ 1 espace)
"." = SPEED WRITE 1:SAVE" ; il suffit d'ajouter le nom du programme et de faire "enter" pour que votre programme soit enregistré en vitesse rapide (nous n'utilisons que cette vitesse).

La petite touche "Enter" n'est pas redéfinie. Il est très pratique d'avoir la commande LIST("3") située juste au-dessus...
Le fait d'avoir déjà prévu le blanc (espace) après certaines fonctions telles que PRINT, LOCATE, EDIT, etc., permet d'entrer immédiatement des caractères sans risque d'un "Syntax Error". Pourquoi avoir défini PRINT alors que nous disposons du point d'interrogation ? Pour ne pas avoir à faire SHIFT...

Cinq touches, normalement très peu utilisées du clavier principal ont également été redéfinies. Le but recherché est de pouvoir taper *d'une seule main,* sans que l'on ait à appuyer sur SHIFT. L'index de cette main libre peut ainsi rester sur le listing du programme à recopier. Non seulement le risque d'erreur de transcription devient presque nul, mais vous serez étonné par votre vitesse de frappe !
— Les crochets deviennent tout bêtement les parenthèses,
— la touche "TAB" devient le signe "=",
— la touche "a commercial" devient le double côte ou guillemets,
— la touche "anti slash" (située au-dessus de la touche CTRL) devient le signe "$".

Dans la pratique, vous n'aurez à faire SHIFT que pour les caractères suivants :

#, %, &, ', *, +, < et >

Avouez qu'ils sont beaucoup moins fréquents que (,), =, " et $.

Résumons :
- Nous partons avec l'AMSTRAD éteint. La cassette contenant ce programme (en début de bande) est toujours "rangée" dans le magnétocassette.
- Mise sous tension — touche PLAY — touches CTRL + ENTER (la petite) — barre d'espacement.
- Il y a alors chargement puis RUN automatique de ce court programme (14 secondes).
- La légende de ces seize touches apparaît à l'écran ; mais on la sait très vite par cœur.

NOTE : Le signe flèche vers le haut rappelle que le ENTER (=CHR$(13)) est inclus.

Pour vous débarrasser de ce programme, tapez la touche ''N'' (ou ''n'') : cela engendre un CLS et un NEW, mais les touches restent définies... Toute autre touche tapée aboutit à un END, ce qui permet alors de lister ce programme.

Nous rappelons que les touches redéfinies par les fonctions KEY ou KEY DEF sont à l'abri des ordres NEW, LOAD, CLEAR, etc. Seule la ré-initalisation de l'appareil par CTRL-SHIFT + ESC peut les annuler.

Le listing ci-dessous (comme tous les autres de cet ouvrage) est imprimé en 40 caractères par ligne, exactement comme sur votre écran en MODE 1.

```
1  ' CHAPITRE 1
10 'PROGRAMME KEY=TOUCHES DE FONCTIONS
20 '  ------ AMSTRAD CPC 464 -------
30 '  -- Non modifiees par NEW --
40 ' Michel Archambault / mars 1985
50  KEY 128,"PRINT ":' 0
60  KEY 129,"LOCATE ":' 1
70  KEY 130,"CHR$(":' 2
80  KEY 131,"LIST ":' 3
90  KEY 132,"INK 0,13:INK 1,0:BORDER 13"
 +CHR$(13):' 4
100 KEY 133,"INK 0,1:INK 1,16"+CHR$(13):
' 5
110 KEY 134, "KEY 134":' 6
120 KEY 135,"CLS"+CHR$(13):' 7
130 KEY 136,"FOR I=1 TO ":' 8
140 KEY 137,"EDIT ":' 9
150 KEY 138,"SPEED WRITE 1:SAVE "+CHR$(3
4):' .
160 KEY DEF 17,0,40:' (
170 KEY DEF 19,0,41:' )
180 KEY DEF 22,0,36:' $
190 KEY DEF 26,0,34:' "
200 KEY DEF 68,0,61:' =
300 'TABLEAU
310 CLS:ORIGIN 0,0:PEN 3
320 LOCATE 10,2:PRINT "PAVE DES TOUCHES-
CLEFS"
330 PEN 1
```

```
340 LOCATE 4,6:PRINT "CLS^","  FOR I=1 TO
 ","    EDIT"
350 LOCATE 2,9:PRINT "Noir/Blanc^","  Ros
e/Bleu^","    libre"
360 LOCATE 4,12: PRINT "LOCATE","    CHR$
(","    LIST "
370 LOCATE 4,15: PRINT "PRINT    SPEED WR
ITE 1:SAVE "
380 LOCATE 12,18: PRINT "TAB est le sign
e ="
390 LOCATE 10,19:PRINT "Crochets = Paren
theses"
400 LOCATE 8,20: PRINT  "A commercial =
Guillemets"
410 LOCATE 14,21:PRINT "anti-slash = $"
420 LOCATE 10,23:PRINT "TAPEZ N POUR NEW
 + CLS"
430 PLOT 5,340:DRAWR 630,0,2
440 DRAWR 0,-190
450 DRAWR -630,0:DRAWR 0,190
460 WHILE R$="":R$=INKEY$:WEND
470 IF R$="N" OR R$="n" THEN CLS:NEW
480 END
490 '---------- FIN DE LISTING ----------
```

Après l'avoir tapé, essayez-le par RUN, mais ne tapez surtout pas la touche N (= NEW...) ! Mettez en place une cassette vierge de bonne qualité (car elle va servir souvent), passez l'amorce, puis pressez le point décimal du pavé numérique. Ajoutez le nom du programme (KEY) et Enter pour le sauver. Deux fois de suite par sécurité.

Nous vous conseillons de ne pas omettre les REM situés aux bouts des lignes 50 à 200 ; ils peuvent être très utiles. Un exemple concret : vous voulez entreprendre de recopier un long listing comportant des DATA d'adresses en hexadécimal ; le signe "$" devenant peu utile, vous allez le remplacer par le "&".

Chargez le programme, barre d'espacement et, après LIST, vous voyez que c'est la ligne 180 qu'il faut modifier. EDIT 180 afin de remplacer 36 (code ASCII de $) par 38 (code ASCII de &). Faites RUN, tapez N, vous êtes prêt.

NOTE : S'il s'agissait d'une fastidieuse suite de POKE avec adresses en hexa, vous pouvez aussi taper ''6'' afin d'avoir :

KEY 134,''POKE &''

Vous n'êtes peut-être pas encore familiarisé avec certaines fonctions Basic utilisées dans ce programme. Rassurez-vous, nous les étudierons en temps voulu.

Pour fixer les idées, ce programme occupe environ 1200 octets. Pour connaître la longueur exacte du vôtre, tapez :

PRINT 43533 – FRE(0)

en sachant que 43533 est le nombre d'octets disponibles à la mise sous tensions.

LA REGLETTE DE TABULATION

Nous ne répéterons jamais assez que l'esthétique des pages d'écran est une chose des plus importantes. Le premier soin consiste à centrer un titre, un mot ou une phrase, et ce au moyen de LOCATE ou TAB. Facile à dire ! En effet, déterminer le positionnement horizontal de la chaîne à centrer est soit trop ''pifométrique'', soit fastidieux à calculer. Grâce à cette réglette que l'on pose sur l'écran, on lit immédiatement le positionnement adéquat, sans le moindre calcul. Un exemple :

Sur la ligne n° 3, nous allons écrire, avec centrage, ''TITRES CENTRES'', tapons seulement :

10 LOCATE,3:PRINT''TITRES CENTRES''

Ne faisons pas encore Enter. Posons la réglette sur l'écran, exactement sur la chaîne à centrer : nous y lisons ''14''. Par la flèche gauche, ramenons le curseur sur la virgule située après ''LOCATE'' et tapons ''14'' puis Enter. Nous obtenons alors :

10 LOCATE 14,3:PRINT''TITRES CENTRES''

Par RUN, on obtiendra un centrage impeccable.

A présent, confectionnons cet outil miracle. Il nous faut seulement une bande de bristol (cartoline) d'environ 25 × 4 cm.

Commençons par le MODE 1. Tapez un BORDER quelconque afin de bien visualiser la fenêtre d'écran. Tapez quatre fois à la file ''1 2 3 4 5 6 7 8 9 0'', ce qui numérote les colonnes de 1 à 40. Bien entendu, le premier zéro correspond à 10, le deuxième à 20, etc...

Placez la réglette sous cette chaîne de chiffres en ajustant soigneusement son extrémité gauche sur le bord gauche de la fenêtre. Recopiez sur la réglette les chiffres qui se trouvent exactement au-dessus sur l'écran (écrire petit).

Cette première échelle va bien sûr permettre de mesurer la longueur (LEN) d'une chaîne, mais là n'est pas le but.

REGLETTE DE TABULATION

MODE 0		MODE 1		MODE 2	
2	10	2	20	2	40
4	9	4	19	4	39
6	8	6	18	6	38
8	7	8	17	8	37
10	6	10	16	10	36
12	5	12	15	12	35
14	4	14	14	14	34
16	3	16	13	16	33
18	2	18	12	18	32
20	1	20	11	20	31
		22	10	22	30
		24	9	24	29
		26	8	26	28
		28	7	28	27
		30	6	30	26
		32	5	32	25
		34	4	34	24
		36	3	36	23
		38	2	38	22
		40	1	40	21
				42	20
				44	19
				46	18
				48	17
				50	16
				52	15
				54	14
				56	13
				58	12
				60	11
				62	10
				64	9
				66	8
				68	7
				70	6
				72	5
				74	4
				76	3
				78	2
				80	1

Sous certaines valeurs de cette échelle, vous allez inscrire en caractères plus gras (stylo feutre) les valeurs correspondant au tableau ci-dessous.
(MODE 1) : Sous le "2" : "20", sous le "4" : "19", sous le "6" : "18", etc.

Voilà pour le MODE 1, le plus souvent utilisé. Retournez la réglette pour les deux autres MODE.

Pour le MODE 0, vous écrivez deux séries à la file de chiffres de 1 à 0 et graduez un bord de la réglette, toujours en vous reportant au tableau ci-dessous. Idem pour le MODE 2 (1 à 80) sur l'autre bord de la réglette.

Ces valeurs ont été obtenues par la formule Tabulation = (NC – LEN)/2 + 1 où NC = nombre de caractères par ligne (20, 40, 80).

On résume : sur une face de la réglette, nous avons MODE 0 et MODE 2, sur l'autre face, MODE 1. L'autre bord de cette face va représenter la numérotation verticale des lignes. Pour cela, tapez la ligne suivante (en mode direct) :

```
CLS:FOR I=1 TO 25:PRINT I : NEXT : FOR I=1 TO 3E5:NEXT
```

Placez votre réglette verticalement sur l'écran en ajustant le bord supérieur sur le haut de la fenêtre, et reporter ces numéros sur le bristol.

NOTE : La seconde partie de la ligne tapée vous laisse très largement le temps de faire ce travail ; sans elle, l'inscription "Ready" nous ferait "scroller" l'écran de deux lignes (interrompue par la touche ESC).
Cette quatrième et dernière échelle nous sera très utile en maintes occasions...

DEUX CAS PARTICULIERS

- La longueur de la chaîne est une valeur impaire et, de ce fait, le dernier caractère "tombe" entre deux valeurs de tabulation, par exemple entre "9" et "8". Comme on ne peut afficher "8.5", choisissez arbitrairement la plus petite.
- La chaîne à l'écran est "à cheval" sur deux lignes. Tapez ceci :

```
10 LOCATE ,12:PRINT"CENTRAGE D'UNE CHAINE LONGUE"
```

C'est là que va intervenir l'échelle "du haut" de 1 à 40.
Mesurez sur la première ligne, qui s'arrête au "I" du mot "chaîne". Notre réglette nous indique que cette longueur est de 19 caractères.

Passons à la ligne inférieure qui commence donc par "N" du mot "chaîne", c'est le 19 + 1 = 20e caractère. On place donc la graduation "20" de l'échelle supérieure sous cette lettre, et sous le "E" du mot "longue", nous lisons la tabulation "7" ; valeur que nous intercalons en amenant le curseur sur la virgule suivant LOCATE. Enter, et la ligne Basic est terminée.

Un dernier conseil : cette précieuse réglette peut s'égarer, s'abimer ; alors faites-en quelques photocopies. Posez-la au-dessus du clavier.

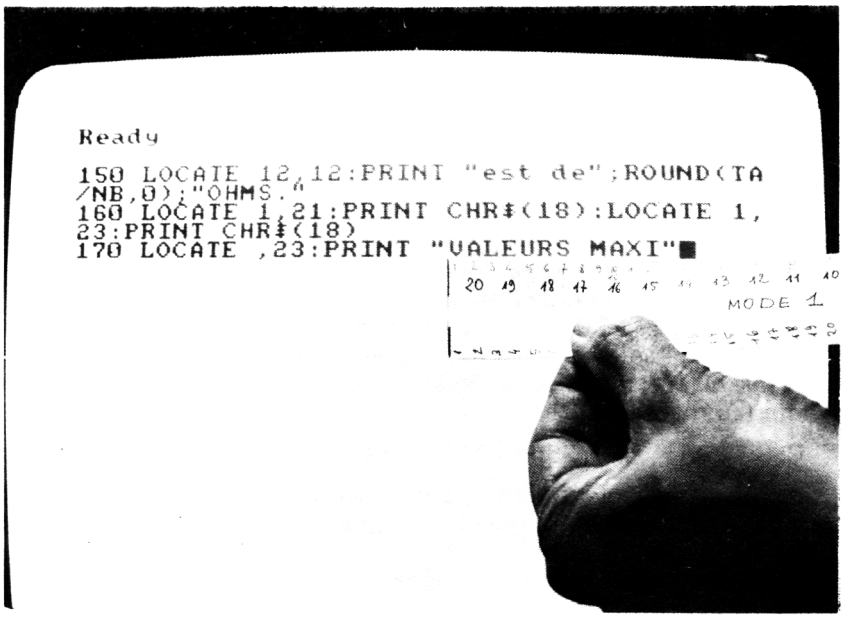


Figure I1 — On place la réglette sous le mot ou la phrase à centrer. Ici, on lit <15>, tabulation à insérer après le LOCATE.

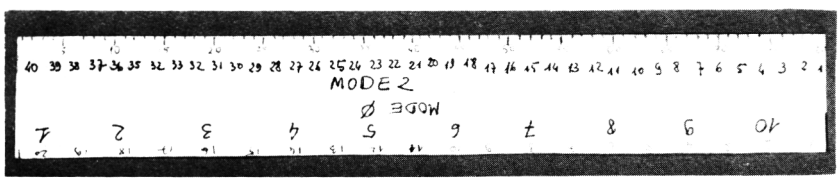
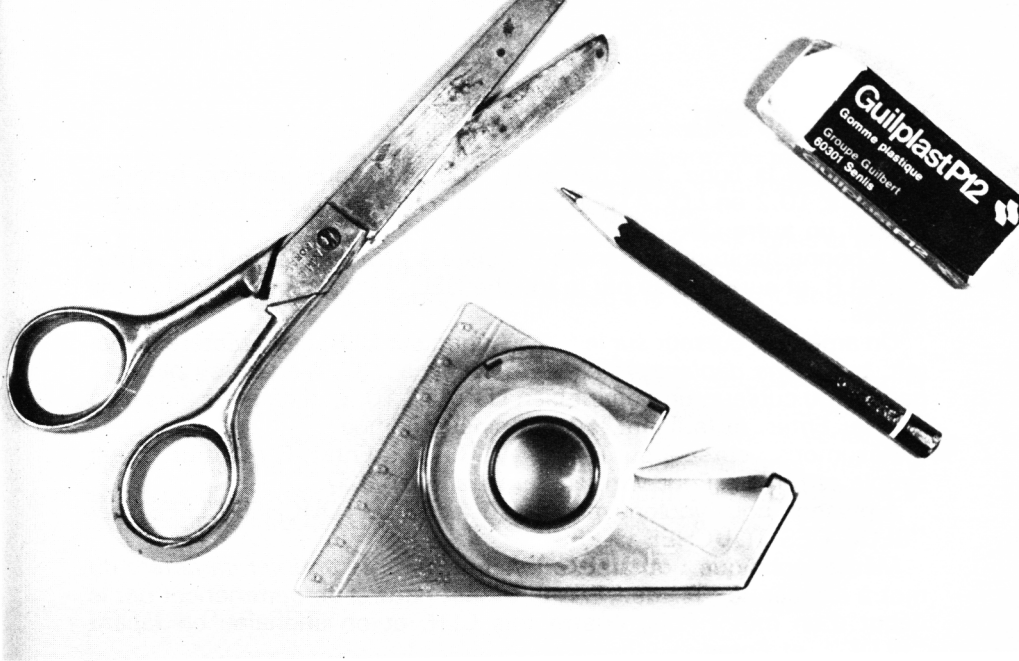


Figure I2 — L'autre face de la réglette de tabulation concerne les MODE 0 et MODE 2.



Chapitre II

LA PRATIQUE DE L'EDITEUR

Dans un micro-ordinateur, on appelle ''éditeur'' le ou les dispositifs dont on dispose pour modifier des lignes de programme. Il en existe de deux sortes : l'éditeur ''pleine page'' où l'on opère directement dans l'écran sur lequel on a fait un LIST, et l'éditeur ''ligne à ligne'' où il faut appeler la ligne à corriger par EDIT. Ici encore, AMSTRAD se singularise puisqu'il possède les deux, mais de manières incomplètes...

Nous disposons en fait de trois procédés d'édition ; par EDIT, par COPY et par SHIFT + COPY. On choisit celui qui s'adapte le mieux à la retouche que l'on veut effectuer. Avantage : notre éditeur est super puissant. Inconvénient : il y a beaucoup de choses à apprendre, et un bon entraînement est nécessaire pour ne pas s'y perdre !

Nous allons expliciter tout cela en essayant d'être plus clair que le manuel d'origine, tout en vous montrant quelques trucs souvent très utiles.

Premières notions à bien retenir : En EDIT, on est toujours en INSERTION de caractères. En COPY, on est en SURIMPRESSION (ou substitution) de caractères. OK ?

Comme programme cobaye, nous allons charcuter le programme ''KEY'', mais en fin de séance il ne faudra pas le sauver sur cassette, ni le lancer par un RUN !

EDITION PAR EDIT

Appelons la ligne 320 par EDIT 320, car nous voulons modifier LOCATE 10,2 en LOCATE 10,3, c'est-à-dire remplacer un caractère (2) par un autre (3).

La bonne habitude à prendre consiste à *effacer d'abord* par la touche CLR, et surtout pas par la touche DEL, génératrice de confusions faciles.

On amène le curseur *sur* le ''2''. Action sur CLR, ce caractère disparaît, puis *sans déplacer le curseur,* on tape ''3''. Venant se placer à gauche du curseur, on obtient bien LOCATE 10,3:. Il ne reste plus qu'à presser Enter, et la ligne 320 est ainsi modifiée.

Résumons : curseur sur le ''2'', CLR, ''3'', Enter. C'est tout ; c'est simple et rapide. Pour vérifier, refaites EDIT 320.

A présent, nous voulons modifier ''PAVE DES TOUCHES-CLEFS'' par ''BLOC DES TOUCHES CLEFS''.

Même technique : on amène le curseur *sur le premier caractère* du mot à effacer, le ''P'' de ''PAVE'', c'est naturel de commencer par le début d'un mot ! Puis, quatre fois CLR, et on enchaîne en tapant ''BLOC'', et Enter.

On complique un peu. On veut remplacer ''BLOC DE TOUCHES CLEFS'' par ''CLEFS DES TOUCHES''. D'abord refaire EDIT 320. Effaçons ''BLOC'', tapons ''CLEFS'' à la place. Deuxième correction : on amène le curseur sur le '' – '', on presse six fois CLR afin de faire disparaître le '' – CLEFS''. Mais, avant de faire Enter, il y a une troisième modification à faire, la tabulation horizontale du LOCATE... Notre réglette nous indique ''12''. Curseur sur le ''0'' du ''10'', CLR, ''2'' et le Enter final.

QUELQUES PETITES REMARQUES

- La ligne éditée occupe plusieurs lignes à l'écran, et votre modification se situe vers las fin. Vous pouvez prendre un raccourci en utilisant la flèche ''vers le bas'' puis la flèche ''vers la droite''.
- Par EDIT, vous avez par erreur demandé une autre ligne : vous êtes *obligé* de taper Enter avant de recommencer (ou faire Break par ESC).
- En cours de modifications, vous vous apercevez que vous vous êtes complètement trompé : n'appuyez surtout pas sur Enter, mais sur la touche ESC, toutes ces bêtises seront oubliées et la ligne éditée ne sera pas altérée.
- Vous voulez *dupliquer* la ligne 320 sans le numéro 1745 : en faisant EDIT 320, le curseur se place directement sur le premier chiffre du numéro de ligne (le ''3''). Il suffit de taper trois fois CLR puis ''1745'' et Enter. Vous venez de créer la ligne 1745 identique à la 320, laquelle existe toujours.
- Vous voulez *changer* le numéro d'une ligne, par exemple, la 320 devenant la 325. Même technique que précédemment pour la dupliquer en 325. Mais ensuite tapez 320 et Enter afin d'effacer la ligne 320.

- Vous voulez *allonger* une ligne : EDIT, curseur sur le dernier caractère, complétez, Enter. A ce propos, vous gagnerez du temps en faisant CTRL + flèche en bas : le curseur se place immédiatement sur le dernier caractère de la dernière ligne d'écran.
- Tapez une ligne 999'AAA..., etc., de quoi remplir trois lignes et demie de caractères quelconques. Enter, EDIT 999 et observez comment se comporte le curseur lorsque l'on presse à la fois CTRL et une touche flèche...

TRES IMPORTANT : Lorsque vous êtes en train de composer une ligne nouvelle, vous êtes exactement dans la même situation qu'en EDIT, c'est-à-dire que vous êtes en insertion, et tout ce que nous venons de décrire pour une ligne affichée par EDIT est applicable pour une ligne en cours d'écriture. Nous l'avons d'ailleurs montré au chapitre précédent pour inclure la tabulation lue sur la réglette après le LOCATE.

EDITION PAR COPY

Nous savons que la ligne à modifier se situe dans les lignes 300, alors entrons CLS puis

LIST 300 – 399

Il faut que la première ligne nommée existe vraiment, la ligne limite (399) est moins exigeante. Les lignes 300 à 390 apparaissent à l'écran.

Voilà ! C'est la ligne 310 où nous voulons remplacer PEN3 par PEN2. Pour une modification aussi simple, on a plus vite fait d'amener le curseur au début de cette ligne (le ''3'' de ''310''), et de la recopier par la touche COPY. Dès que le curseur est sur le ''3'' à changer, on tape ''2'' à sa place, et Enter, c'est tout.

Traitons la ligne 340 afin de transformer le ''LOCATE 4,6:'' en ''LOCATE 4,5:''. Même manœuvre quand COPY nous amène le curseur sur le ''6'', on tape ''5'', mail il faut continuer par COPY *jusqu'à la fin de la ligne !* Si vous aviez fait Enter sitôt la correction faite, vous auriez eu la ligne :

340 LOCATE 4,5

d'où quelques surprises au lancement...

Vous venez de commettre cette étourderie, mais vous l'avez constaté aussitôt : rien n'est perdu puisque la ligne 340 (en entier) est toujours écrite à l'écran. Ramenez le curseur sur le début de ligne et repassez-la *entièrement* par COPY, puis Enter ; la bêtise est réparée. Ouf !

Si, par COPY, le curseur a dépassé le caractère à substituer, vous pouvez revenir en arrière par DEL et corriger. C'est risqué et pas très propre car DEL laisse derrière lui un carré couleur PEN. En cas de doute, n'hésitez pas à annuler par ESC, mais la ligne à l'écran est ''polluée'' par un ''*Break*''. Rappelez alors la ligne en EDIT.

On peut faire bien d'autres choses par COPY, même des insertions, mais ceci est très acrobatique, donc aléatoire. Sur le plan pratique, n'utilisez COPY *que pour de simples substitutions de caractères.* C'est alors plus simple et plus rapide que par EDIT.

EDITION PAR SHIFT + COPY

Cette technique est un peu plus ''lourde'' que la précédente, mais elle est plus sécurisante. En outre elle permet de réaliser des ''charcutages'' de lignes, tels les fusions et scindages, qui sont irréalisables sur la plupart des autres micro-ordinateurs. Tapez CLS puis :

LIST 400 – et Enter

c'est-à-dire de la ligne 400 à la fin.

Le curseur se place en bas du listing sur la ligne 400. Vous remarquez alors que vous avez *deux* curseurs ; l'ancien, en bas de l'écran, et celui que vous venez d'amener.

Relâchez SHIFT.

Pressez COPY : la ligne 400 se recopie à partir de l'ancien curseur (en bas d'écran). Terminez la ligne sans faire Enter : vous constaterez que vous pouvez la rallonger par des caractères quelconques, revenir en arrière par la flèche ''vers la gauche'' et faire des insertions. Exactement comme si l'on avait tapé EDIT 400. Quel intérêt alors ? Le ''charcutage contrôlé'' !

Faites Break par ESC.

L'ABLATION DE CARACTERES

Nous voulons, dans la ligne 400, supprimer ''A commercial ='' : avec SHIFT et flèche, remontons à 400. On commence la copie par la touche COPY (SHIFT peut être enfoncé ou non) jusqu'à PRINT''. A cet endroit, lâchons COPY et faisons SHIFT + flèche à droite jusqu'à dépasser le signe ''='' . Vous remarquerez que le curseur du bas, lui, n'a pas bougé. Puis continuons par COPY. En bas s'inscrit ''guillemets''.

Toutefois, avant de faire Enter, vous constatez qu'il y a dans votre future ligne 400 des blancs superflus : revenez en arrière par la flèche à gauche (sans SHIFT) et CLR pour les effacer, puis de là Enter. Vous pouvez vérifiez votre travail par EDIT 400.

Ce qu'il faut retenir : Lorsque SHIFT est enfoncé, on peut promener le ''curseur du haut'' n'importe où dans l'écran, le ''curseur du bas'', lui, attend qu'on appuie sur COPY (ou qu'on entre des caractères nouveaux au clavier). La ligne qui s'écrit en bas est bien celle qui résulte de toutes nos manœuvres.

A partir de ces concepts, on va pouvoir faire des miracles !

FUSION DE LIGNES

Nous voulons réunir les ''clauses'' des lignes 400 et 450. On appelle clauses les fragments de lignes séparés par des '':'' ou le contenu d'une ligne courte. Il y a, par exemple, trois clauses dans la ligne 460 et une clause dans la ligne 440.

Par SHIFT, remontez le curseur sur la ligne 400. COPY sur celle-ci. Tapez '':'' au clavier (ce caractère ne s'inscrit *que* sur la ligne du bas).

Par SHIFT + flèche, positionnez le curseur du haut sur le début de la première clause de la ligne 450, donc sur le ''D'' de ''DRAWR – 630,0''. Puis COPY jusqu'en fin de ligne. Vous remarquez alors que la ligne 440 du bas comporte bien les trois clauses bout-à-bout. Faites Enter.

Ensuite, n'oubliez pas de taper 450 et Enter. Refaites CLS et LIST 400.

SCINDAGE DE LIGNE

Opération inverse, nous allons faire deux lignes (430 et 435) de la ligne 430 :
SHIFT + flèche sur la 430 ; COPY sur la première clause (sans prendre le '':'') et Enter.

Tapez 435. Remontez en SHIFT + flèche sur le départ de la seconde clause de la ligne 430 (le D de DRAWR) ; COPY, Enter ; mission accomplie !

Retenez bien de ces deux derniers exemples que, dès que l'on relâche COPY, la ''ligne future'' (celle du bas) prend en compte ce que vous tapez au clavier principal. Dès qu'on reprend COPY, il y a duplication des caractères se trouvant sous le ''curseur du haut''. En ce qui concerne les déplacements des curseurs par les touches flèches : sans SHIFT = curseur du bas. Avec SHIFT = curseur du haut. Facile à retenir !

PERMUTATION DE NUMEROS DE LIGNES

Pour inverser les lignes 480 et 490 : tapez 480, montez sur la clause de la ligne 490, COPY, Enter. Tapez 490, montez sur la clause de la ligne 480, COPY, Enter.

A noter que ce serait impossible par EDIT. Par la même technique, on pourrait aussi réécrire une ligne à plusieurs clauses afin de permuter celles-ci. Faites un essai avec la ligne 310.

TEST EN MODE DIRECT

Tapez CLS et LIST 400 – . Remontez en SHIFT + flèche sur le début de la clause (sans le numéro de ligne) de la ligne 420 : COPY, Enter.

Le contenu de la ligne 420 est exécuté. C'est très commode pour vérifier l'action d'une ligne Basic.

LE DOUBLE ECRAN

Vous n'avez pas d'imprimante, et vous voulez travailler dans une zone de programme tout en vous inspirant d'une autre zone hélas très éloignée dans le listing : séparez votre écran en deux fenêtres ! Tapez :

WINDOW # 0,1,40,1,12:CLS:WINDOW # 1,1,40,3,25:CLS # 1.

NOTE : N'oubliez jamais le CLS après avoir défini une fenêtre. Tapez ensuite :

LIST – 100 (jusqu'à 100)

puis LIST 400 – , # 1 puis LIST 120 – 160. Essayez alors de faire passer le curseur dans la fenêtre du bas : IMPOSSIBLE. Tapez alors LIST 300 – 350, # 1. Le curseur reste toujours bloqué sur la fenêtre du haut.

En conséquence, zone modèle en bas, zone de travail en haut.

Pour revenir à la fenêtre unique, tapez :

WINDOW 1,40,1,25

CONCLUSION

Comme nous vous l'avons démontré, l'éditeur de l'AMSTRAD est très puissant, mais pour l'exploiter il faut s'y exercer régulièrement pour l'avoir ''bien en tête''.

Ne vous dites surtout pas ''je n'aurai jamais besoin de toutes ces pirouettes''. Erreur !Quand votre programme dépassera les cent lignes et que vous sentirez le besoin de faire du ''rangement'', du ''ménage'' et de l'amélioration, vous comprendrez qu'il est bien plus agréable de jouer du curseur cavaleur que de retaper une multitude de très longues lignes...

Les sujets traités dans ces deux premiers chapitres étaient vraiment très terre-à-terre, mais nous estimons que ces notions de base doivent être parfaitement acquises dès le départ. En effet, vous vous apprêtez à écrire, à concevoir des programmes risquant d'être longs ; il faut donc être en mesure de ne pas perdre du temps en les tapant ou en les retouchant. Vous pouvez éteindre momentanément votre appareil, car les trois chapitres qui suivent vont ''voler un peu plus haut''. Mais, rassurez-vous, ils ne seront pas plus difficiles...

Chapitre III

LA PROGRAMMATION DITE STRUCTURÉE

Un programme ''structuré'' ou encore ''modulaire'' se divise en deux parties distinctes :

— le programme principal, assez court, constitué essentiellement d'une suite de GOSUB et se terminant par un END,

— le bloc de sous-programmes (ou modules) appelés par tous ces GOSUB. Chaque module commence par un REM (') pour le titre, et se termine par un RETURN. L'ordre de ces modules n'a aucune espèce d'importance.

Cette configuration peut vous sembler bizarre, mais nous vous montrerons que c'est une panacée, que des avantages sans le moindre inconvénient.

A l'inverse du structuré, il y a le ''programme fleuve'' qui commence à la première ligne et se termine au bas du listing. Le programme ''KEY'' est de ce type, mais ce n'est pas gênant, puisqu'il est à la fois COURT et SIMPLE.

Les petits programmes figurant dans le manuel sont courts, eux aussi, et de ce fait sont en fleuve. C'est normal, ils servent à illustrer une fonction Basic, trois au maximum. On y rencontre parfois un GOSUB, mais c'est pour éviter de retranscrire plusieurs fois le même petit groupe de lignes. De ce fait, le débutant en Basic croit que GOSUB n'est utilisé que pour les répétitions ; c'est aussi bien autre chose : le concepteur utilise GOSUB lorsqu'il s'attaque à une *étape précise* de son programme ; ce module, ainsi appelé, peut faire deux lignes au départ, et

quinze en final ! Autrement dit, au lieu de tout souder bout-à-bout, on prend la sage précaution de prévoir des pièces démontables, interchangeables ; donc un ensemble très souple parce que *très facilement modifiable.*

Se lancer dans un long programme fleuve, est une chose suicidaire : vous aboutirez à un diplodocus truffé de GOTO vers le haut et le bas, c'est ce qu'en jargon nous appelons un ''programme spaghetti'' ou ''programme bouts de ficelles'', car on ne sait par quel bout le prendre...

Or, voici les deux grandes lois de l'informatique :

1 — Un programme ne fonctionne jamais du premier coup.
2 — Un programme parfaitement au point n'est jamais définitif.

A noter que ces deux lois s'appliquent à tous les langages, en micro, en mini et en grosse informatique.

Quand la loi n° 1 ''tombe'' sur un programme spaghetti, son malheureux auteur est confronté au plus éprouvant des jeux d'aventure ; ces IF... THEN... GOTO... n'ont rien à envier aux salles du château maudit sur cassette du commerce.

Bravo ! Le héros est enfin sorti du labyrinthe BASIGOTO, et son programme fonctionne.

Quelques mois plus tard, c'est la loi n° 2 qui se présente ; une petite amélioration à effectuer. Il reste à trouver la bonne entrée du labyrinthe et le jeu (?) recommence...

Avec un programme structuré tout se passe très différemment, car, sans être fin limier, on détermine très vite s'il s'agit d'une erreur d'assemblage (programme principal) ou d'une pièce défectueuse (un des sous-programmes). La réparation est alors propre et surtout rapide. Idem pour les modifications, quitte à créer un module supplémentaire.

Ecrire un programme fleuve de plus de 3000 octets est déjà de la haute voltige, et peu de programmeurs professionnels s'y risqueraient... Alors, faites comme l'auteur, ne mettez pas les pieds dans ce genre de galère.

Après ce réquisitoire très partial, étudions les choses de plus près.

LA CONSTITUTION D'UN SOUS-PROGRAMME

Si votre programme principal se termine à la ligne 320, ne commencez pas votre bloc de sous-programmes en 330 ! Mais carrément en 10000 (5000 à la rigueur).

Chaque module débute à une ligne multiple de 1000. Exemples : 10000, 11000, 12000, etc. Comme vous pouvez aller jusqu'au numéro 64000, vous n'aurez jamais 55 modules à écrire...

Cette ligne en mille est toujours un REM. Exemples : 10000 'TITRE ; 11000 'CALCUL DE LA DUREE DR ; etc. Afin d'améliorer la lisibilité du listing, vous ajoutez des ''–'' ou des ''*'' avant et après l'intitulé. Exemple : 17000 ' – – – – – ENTREE DE NM$ – – – – –.

Le contenu du module peut être très court ; c'est souvent une seule ligne pour une complexe formule de calcul. Toutefois, il est prudent de toujours mettre l'instruction RETURN seule sur une ligne, et avec un numéro de ligne assez éloigné de la ligne précédente, cela au cas où on aurait à intercaler d'autres lignes.

Pour le numéro de la ligne RETURN pressez de préférence un multiple de 100 (la clarté...).

Il est banal qu'au sein d'un sous-programme on trouve d'autres GOSUB vers d'autres sous-programmes. C'est pratique est sans danger.

Rappelons que, ces modules étant indépendantes, leur ordre dans la numérotation n'a aucune importance.

Une seule précaution est à respecter :
On ne doit jamais sortir définitivement d'un sous-programme autrement qu'en passant par sa ligne finale comportant le RETURN. Si une condition IF autorise une sortie prématurée, n'écrivez surtout pas :

```
IF R=0 THEN RETURN, mais plutôt
IF R=0 THEN 16400.
```

16400 étant la ligne de *l'unique* RETURN.

A plus forte raison ne sortez pas d'un sous-programme par un GOTO, ce serait le suicide assuré...

Il est très fréquent que l'on écrive un module en sachant très bien que le programme n'y passera qu'une seule fois, et encore à la suite d'un IF... THEN GOSUB... assez rarissime (une sécurité). Plus tard, en peaufinant le programme principal, il n'est pas rare que l'on trouve une deuxième utilisation de ce groupe de lignes (en ''fleuve'', il aurait fallu les réécrire...).

Mieux encore : en écrivant ce sous-programme, vous réalisez qu'il peut vous servir pour des programmes futurs. C'est ce qu'on appelle un UTILITAIRE ; il lui faut un régime de faveur qui consiste à lui donner un numéro de ligne à partir de 50000. Voilà le détail de cette astuce :

Quand vous aurez terminé votre gros programme, qu'il sera sauvé sur cassette, vous ferez DELETE – 49999. Il ne restera plus en mémoire que l'ensemble de ces précieux utilitaires, que vous sauverez sur une cassette séparée sous le nom ''UTIL''. Pour le gros programme suivant, on charge ce programme avant ; ou après en faisant :

```
MERGE '' ''
```

C'est une pratique utilisée systématiquement par l'auteur. Que voulez-vous, dans l'AMSTRAD on a prévu des touches et des fonctions pour les fainéants impatients, alors autant s'en servir...

LE PROGRAMME PRINCIPAL

Rassurez-vous, il n'y a pas que des GOSUB ! Il y a beaucoup de lignes d'instructions Basic, mais assez clairsemées car les GOSUB sont en majorité, d'autant plus que l'on ne met généralement *qu'un seul* GOSUB par ligne. Motif : c'est plus facile à modifier, et cela permet de mettre un REM explicatif en bout de ligne.

Et oui ! Beaucoup de REM, s'il vous plaît, la colossale mémoire de l'AMSTRAD nous le permet. En effet, sans eux, cette sèche succession ne serait guère explicite. Ne dites pas ''ce n'est pas pour être publié dans une revue, c'est pour moi'' ! Quelle bévue ! Car, lorsque dans quelques mois vous reprendrez ce listing pour l'améliorer ou pour vous en inspirer, vous constaterez que votre mémoire est bien décevante : vous perdrez davantage de temps à comprendre ce que vous aviez écrit que vous n'en auriez mis pour taper quelques REM çà et là.
Pour ces REM, deux cas se présentent :

— les modules sont nombreux et ne sont généralement appelés qu'une fois : mettre le REM sur la ligne du GOSUB ;
— il n'y a qu'une demi douzaine de modules, mais ils sont utilisés fréquemment : faites une légende groupée en début du programme, une fois pour toutes. Exemple :

```
50' 10000=TITRE
60' 11000=CALCUL DE LA DUREE
70' 12000=VALIDATION DE NB
```

etc., mais dans l'ordre des numéros.

Ne dites pas ''pourquoi légender mes modules dans le programme principal puisque chacun d'entre eux débute par cette même légende ?''. Réponse : pour ménager votre équilibre nerveux le jour où vous vous replongerez dans ce programme...

Dernière obligation : terminez le programme principal par un END. Ne l'oubliez pas, sinon il enchaînera sur le premier sous-programme... Cette étourderie est classique.

POUR VOUS RASSURER SUR LES GOSUB

Certaines personnes pensent qu'un GOSUB provoque un ralentissement de l'exécution en raison de cet aller et retour. C'est vrai, mais savez-vous de combien ? De 0,6 millième de seconde...

Afin de chronométrer cela, nous avons lancé ces deux très courts programmes qui font pourtant exactement la même chose, le premier en ''fleuve'', l'autre avec GOSUB.

```
10 'RETARD D'UN GOSUB
20 FOR I=1 TO 2000
30 X=I*15
40 NEXT
50 PRINT CHR$(7);'BIP
60 'DUREE 6.6 SECONDES
70 END
90 '-----------------------
```

```
100 FOR I=1 TO 2000
110 GOSUB 40000
120 NEXT
130 PRINT CHR$(7)
140 'DUREE 7,8 SECONDES
150 END
40000 X=I*15
40100 RETURN
```

Deux mille GOSUB ont occasionné un retard cumulé de 1,2 seconde, soit 1,2/2000 = 0,6 ms par GOSUB. D'autre part, si vous remplacez ce 40000 par 200 ou par 64000, vous obtiendrez toujours 7,8 s. Alors ! Convaincu ?

CONCLUSION

La nécessité de la programmation structurée apparaît à partir d'une taille moyenne de programme. Nous obtenons un ensemble clair, accessible et se prêtant avec souplesse aux plus grandes modifications. D'autre part, certains modules sont directement réutilisables pour d'autres programmes, sans avoir à les recopier.

Prendre l'habitude de penser une conception de programme en structuré, constitue en outre un atout pour ceux qui voudraient, par la suite, apprendre d'autres langages. En effet, la plupart d'entre eux ne peuvent être écrits qu'en structuré uniquement, exemple le FORTH.

Chapitre IV

L'ORGANIGRAMME

C'est un dessin qui représente très schématiquement le plan, la stratégie, d'un programme. On dit aussi ''ordinogramme'', mais c'est plus rare ; à ne pas confondre avec ''algorythme'' qui, lui, est un texte décrivant le scénario du programme, donc de l'organigramme.

Certains ''puristes'' disent qu'il faut toujours tracer l'organigramme avant de taper la première ligne de programme. N'exagerons rien, cela n'est vrai que dans certains cas. C'est le même problème pour un bricoleur qui fera un plan détaillé avant de construire une armoire ou un escalier, mais qui s'en dispensera pour une simple étagère...
C'est un type de dessin qui a ses *normes* tout comme le dessin industriel, mais celles-ci sont fort peu nombreuses, donc vite apprises.

Ce sont le plus souvent les conditions IF en cascades qui rendent les organigrammes complexes. Cela fait autant de ramifications, de voies différentes, qui se ramifient à leur tour, qui se rejoignent, qui se rebouclent. Et il faut que tout cela tienne la route ! Cela rapelle un peu un réseau ferroviaire de centre de triage avec tous ces aiguillages.

C'est un jeu de piste qui est souvent passionnant où il est inutile de bien dessiner, car on remplit la corbeille de feuilles froissées ! La longueur d'un programme n'est pas toujours en rapport avec sa complexité, en voici deux exemples personnels.

• Un problème super complexe a été résolu par un programme de 6000 octets seulement (cinq fois le programme KEY), mais la conception de l'organigramme avait préalablement demandé 35 heures d'étude...

• Un programme de calculs d'électronique a rempli 11300 octets, son organigramme avait été élaboré en moins d'une heure...

Dans le premier exemple, vous entrevoyez qu'il eut été irréaliste de passer au clavier sans ''calculer son coup'' sur le papier.

Avant de concevoir un programme, on griffonne un peu quelques ''stratégies'' sur le papier, le plan d'action. C'est généralement là que l'on détermine le degré de difficulté, s'il faut s'installer pour écrire beaucoup ou si l'on peut déjà mettre l'ordinateur sous tension.

Nous savons très bien que le débutant n'éprouve vraiment aucun attrait pour ces carrés et ces losanges reliés par des flèches, mais sachez que l'on raisonne beaucoup plus efficacement sur un organigramme que sur un listing. Vous gagnerez du temps à le dessiner car vous perdrez beaucoup moins de temps en retouches du programme. Donner un coup de gomme va plus vite que de modifier une ligne... Pardonnez ce long préambule, mais il fallait bien vous motiver.

LES NORMES DU DESSIN

1) Un organigramme se dessine du haut vers le bas. Des traits fléchés peuvent remonter.
2) Chaque étape se symbolise par un rectangle (ou pavé) dans lequel on écrit *très sommairement* ce qui s'y passe. S'il s'agit d'une entrée de donnée (INPUT, INKEY$, etc.), inscrivez ''IN'' dans l'angle en haut à gauche.
3) Une étape conditionnelle (un IF) se représente par un losange ou un hexagone très allongé pour pouvoir y écrire davantage.
4) Ces pavés et losanges sont reliés par des traits fléchés, ce sont les ''trajets'' du logiciel.
5) Quelques figures moins fréquentes (et pas toujours respectées) : un cercle ou un rectangle aux angles arrondis = le début et la fin (END) du programme. Un rectangle dont le côté droit est incliné = affichage des résultats sur imprimante ou sur écran (parfois c'est un rectangle avec l'angle supérieur droit coupé).

Ce sera tout ; il existe d'autres normes, mais que pratiquement personne ne respecte...

Un organigramme est fait pour être pratique, et pour cela il faut, et il suffit, qu'il soit clair, et tout d'abord avec soi-même.

Soyez extrêmement succinct pour vos légendes dans vos pavés. Ecrivez par exemple ''Calcul moyenne M'', mais pas la formule !

Ne mettez pas deux choses différentes dans le même pavé du genre ''Entrée de NB et calcul du total'', mais faites deux pavés distincts reliés par un trait fléché. C'est bien plus prudent car, par la suite, un autre trait fléché peut arriver entre eux deux.

Pour le premier jet, dessinez au crayon afin de pouvoir gommer. Ne vous appliquez pas car le premier dessin va presque toujours à la corbeille à papier.

Le contenu d'un pavé peut très bien correspondre à une seule ligne Basic. Exemple : un calcul. Par contre, il peut aussi correspondre à un sous-programme de cinquante lignes qui appelle deux autres sous-programmes. Ne compliquez pas votre dessin pour autant ; pour ce pavé, prévoyez un petit organigramme-détails sur une feuille séparée. En effet, l'organigramme ne doit représenter que l'interconnexion des différentes étapes, c'est une vue d'ensemble, sans plus, et c'est bien suffisant !

UN EXEMPLE CONCRET

Voici un petit programme qui vous permettra de calculer vos consommations d'essence ou de gas-oil. L'auteur a d'abord conçu l'organigramme, puis a tapé le programme, enfin il a rajouté les numéros de lignes près de chacun des pavés.

Certains passages du listing peuvent encore vous intriguer, rassurez-vous, nous en parlerons ultérieurement. Disons quand même que PRINT CHR$(18) efface la ligne *à droite* du curseur.

```
10 'CONSOMMATION DE CARBURANT
20 CLS:BORDER 9
30 LOCATE 5,2:PRINT "Pour un Plein mettr
e la lettre P"
40 LOCATE 5,4:PRINT "apres les litres.Ex
emple: 23.5P"
50 LOCATE 5,6:PRINT "Pour finir mettre F
 pour litres."
60 PRINT STRING$(40,"_")
70 LOCATE 8,10:INPUT "Kilometrage depart
: ",KD$
80 KD=VAL(KD$):IF KD=0 THEN 70
100 'ENTREE LITRES L
110 LOCATE 8,13:PRINT CHR$(18);:INPUT "L
itres:";L$
120 L=VAL(L$):D$=RIGHT$(L$,1)
130 IF UPPER$(L$)="F" THEN CLS:END
140 IF L=0 THEN 100 ELSE T=T+L
150 IF UPPER$(D$)<>"P" THEN 100
160 LOCATE 23,13:PRINT CHR$(18);:INPUT"K
m:";KA$:'Km Arrivee
170 KA=VAL(KA$):IF KA=0 OR KA<=KD THEN 1
60
```

```
180 GOSUB 10000:'CALCULS
190 GOSUB 11000:'AFFICH.RESULTATS
200 KD=KA:T=0
210 LOCATE 8,10:PRINT "Kilometrage depar
t:";KD
220 GOTO 100
9990 '-----------------------------------
10000 'CALCULS
10010 DIST=KA-KD
10020 CONS=(INT(1000*T/DIST))/10
10100 RETURN
11000 'AFFICHAGE
11010 LOCATE 2,21:PRINT CHR$(18);
11020 PRINT "Consommation:";CONS;"L/100
sur";DIST;"Km"
11100 RETURN
```

Ce programme ne faisant que 900 octets, nous avons pu faire un organigramme assez détaillé. Dans la pratique, on n'y fait pas figurer ces sécurités ''anti-plantages'' que constituent les lignes de validations d'INPUT. Elles sont sous-entendues.

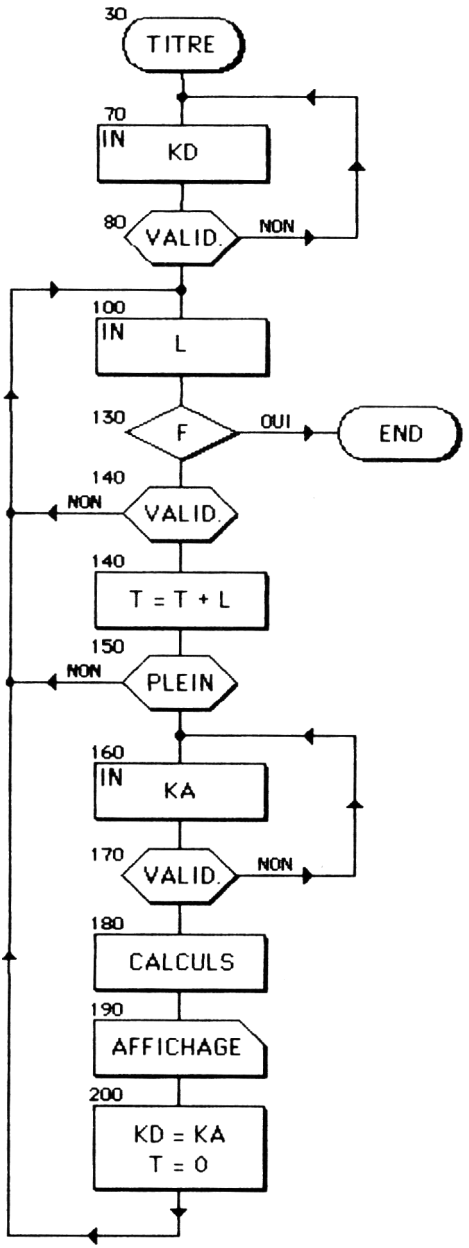
Imaginez que, par la suite, vous vouliez perfectionner ou gadgétiser ce programme ; c'est son organigramme qu'il faut étudier, pas le listing. On voit alors les branchements que l'on peut faire et à quels endroits exacts, grâce aux numéros de lignes.

A ce propos, remarquez que pour les pavés ''calculs'' et ''affichage'' nous avons noté la *ligne qui appelle* ces sous-programmes. C'est plus logique que 10000 ou 11000. On aurait pu aussi fusionner ces deux pavés en un seul appelé ''résultats''.

Nous vous conseillons de taper ce petit programme pour plusieurs raisons : d'abord parce qu'il vous sera utile pour suivre l'état de votre véhicule, ensuite pour constater de visu l'effet de certaines lignes ou de certaines fonctions Basic de l'AMSTRAD généralement méconnues. Vous noterez encore que les petits programmes que nous vous livrons sont souvent des ''produits finis'', avec sécurités sur les entrées et pages d'écran centrées ; donc rien à voir avec les programmes expérimentaux ou de démonstration qui, eux, occuperaient beaucoup moins de lignes.

LE COTE ESPERANTO

Décortiquez point par point ce petit organigramme de la figure IV1. Ensuite, établissez l'organigramme d'un petit programme que vous aviez déjà conçu. Pourquoi cet entraînement ? Parce que la pratique de la lecture de ces dessins va vous être très utile dans le futur :



KD	Km départ
KA	Km arrivée
F	Fin
L	Litres
T	Total litres
VALID.	Validation

Figure IV 1
L'organigramme du programme ''Consommation de carburant'' : le mécanisme apparaît clairement. Les numéros de lignes facilitent la localisation.

Dans une revue de micro-informatique, vous trouvez un programme qui vous fait bien envie. Hélas, il est écrit pour une autre machine dont le Basic est intraduisible en Basic AMSTRAD. Heureusement, il y a l'organigramme, et c'est à partir de lui seul que vous reconstituerez ce programme. En effet, un organigramme ne représente qu'un *enchaînement d'idées,* donc directement traduisible en n'importe quel langage.

Chapitre V

NUMEROS DE LIGNES ET VARIABLES

A ce chapitre, on pourrait donner le sous-titre ''Préparons notre avenir'', car il est évident que le programme sur lequel vous travaillez va ensuite devenir une *archive.* Une archive est une vieillerie qui est *exploitable ;* pour y ''repomper'' un passage dont on était fier ou pour concevoir un programme tout nouveau qui se rattache au même sujet.

Les REM et l'organigramme, même tracé après coup, font beaucoup pour la clarté du listing, mais méfiez-vous de l'anarchie, imposez-vous vos propres normes pour une foule de menus détails. Ainsi, lorsque vous lirez un de vos listings vieux de deux ans, vous arriverez à vous relire sans difficulté. L'auteur, à ses débuts, ne s'était pas méfié de cela, et il le regrette amèrement aujourd'hui.

Nous avons dit ''normes'', il faudrait dire ''habitudes personnelles''. Etablissez-les est respectez-les. Ne les modifiez que pour apporter une amélioration dans votre technique de programmation. Ces détails concernent surtout les numéros de lignes et les noms des variables utilisées. Ces deux points peuvent paraître futiles, l'expérience nous montre tous les jours qu'ils sont des aides précieux.

Répétons que les indications qui vont suivre sont à considérer comme étant seulement des exemples concrets et réels ; libre à vous de les respecter ou de créer les vôtres. Beaucoup des ces ''habitudes'' ont d'ailleurs été calquées sur celles de quelques collègues, lesquels les tenaient de Dieux sait qui...

LA NUMEROTATION RATIONNELLE DES LIGNES

On commence à 10 : un REM contenant le titre et la date. Date à tenir à jour après chaque retouche, sinon comment connaître la dernière version ? Après, d'autres REM si nécessaire.

La fonction AUTO est bien pratique, surtout pour les étourdis. En effet, elle évite de taper deux fois de suite le même numéro (annulant donc la ligne précédente), et vous prévient par un ''*'' qu'il y a encore un programme en RAM. Par contre, il ne faut pas vous laisser piéger par sa triste continuité. Aérez vos numéros en tapant Enter à vide, surtout quand vous abordez un passage scabreux, il sera alors facile d'utiliser ces numéros laissés libres.

Pour mieux séparer vos étapes, vous décidez de passer de la ligne 630 à 1000 : Break, puis tapez AUTO 1000 et poursuivez.

Trois rappels : Les numéros peuvent aller jusqu'à 65535, soit au pas de 10, 6553 lignes... C'est impensable et impossible (la RAM...), donc, ne soyez pas ''radin'', aérez ! D'autre part, sachez qu'un micro-ordinateur met exactement le même nombre de microsecondes pour passer d'une ligne à la suivante, et ce que leurs numéros soient séparés de 1 ou de 65000 unités.

Pour taper le numéro de la ligne 2, il ne faut qu'un caractère ; pour le n° 65530, il en faut cinq. Mais, dans la mémoire, 2 ou 65530 occupent chacun deux octets, pas plus, pas moins.

Vous n'avez donc aucun motif pour ne pas aérez vos numéros de ligne.

Lorsque vous entamez une étape, si petite soit-elle, commencez par un multiple de 100. Ça fait plus ''propre'' de faire un GOTO 600 plutôt qu'un GOTO 530. Si l'étape est un ''gros morceau'', prenez un multiple de 1000. C'est pour cela aussi que nos sous-programmes commencent en 10000, et tous les 1000.

Certains programmes possèdent un ''menu principal'' proposant diverses options : quand l'une est terminée, il y a retour automatique à ce menu, sauf pour l'option FIN qui lance un CLS:END. Dans tous nos programmes, cette étape menu commence à la ligne 1000 parce que c'est facile à retenir. Dans un listing, si on lit GOTO 1000, on sait ce que cela provoque. D'autre part, si le programme ''plante'', ou si l'on arrête volontairement par la touche ESC, on pourra le relancer sans perdre les variables en RAM en tapant :

GOTO 1000 et Enter

même sans avoir le listing sous la main. C'est notre ''après Break'' universel.

NOTE : Il n'est pas rare que cette ligne soit de la forme :

1000 GOSUB 10000:'MENU

Abordons quelques cas particuliers :

— Vous devez intercaler des lignes entre les numéros 510 et 520. Faites 512, 515 et 517, mais jamais deux numéros qui se suivent ; on ne sait jamais...
— Vous voulez intercaler une ligne *provisoire.* C'est une pratique très courante lorsque l'on se débât avec un programme qui boîte. Cette ligne arrête le programme et affiche certaines variables en cours. C'est très efficace pour dépister le ''bug'' (erreur) fantôme. Mais attention ! Le grand danger classique est d'oublier d'effacer cette ligne... Comment la retrouver, la repérer tout de suite parmi tant d'autres si son numéro se termine par 0 ou 5 ? Voilà l'astuce :

839 PRINT N,I:STOP

Un numéro se terminant par 9, ça se remarque ! Ça jure dans un listing.
— Vous créez une petite variante du petit sous-programme en 13000. Vous pouvez déroger en le mettant en 13500 (une sorte de 13000 bis). C'est plus clair que de l'avoir mis en 21000.

LA FONCTION RENUM

Sans rien préciser d'autre, elle va tout renuméroter au pas de 10 et à partir de 10. On est toujours surpris par sa rapidité.

La première règle est de l'utiliser que lorsque le programme est au point, car la ligne 11000 peut devenir 430... Il n'y a plus d'espaces de sécurité.

On opère en plusieurs reprises :

1) Un RENUM simple.
2) On part un peu plus bas pour recréer nos débuts d'étapes en multiples de 100 et de 1000. Par exemple, notre chère et ancienne ligne 1000 est devenue 160.

RENUM 1000,160 et Enter

plus loin, la 2000 est devenue 1240

RENUM 2000,1240, etc.

et ainsi jusqu'en ''bas'' du listing. Fastidieux ? Mais pas du tout, grâce à la touche 6 du pavé numérique

KEY 134,''RENUM''

C'est joli, une numérotation de listing sans le moindre ''défaut''.

LES NOMS DE VARIABLES

Un des grands points forts de l'AMSTRAD concerne le nombre maximum de caractères pour un nom de variable : 40 ! alors que pour la

plupart de ses concurrents, on est malheureusement limité à deux caractères. C'est d'ailleurs à cause de cela que chaque mot Basic doit être limité par un blanc. On a donc le droit de nommer des variables ''TOTAL'', ''ABSENCE'', ''DIMENSION'' que d'autres machines décomposeraient en TO TAL, ABS ENCE, DIM ENSION, d'où des ''Syntax Error'' à la pelle !

Autre point fort : l'AMSTRAD ne fait pas de différence entre majuscules et minuscules en ce qui concerne les noms de variables. Exemples :

ch$ = ''essai'' :PRINT CH$ → essai
NB = 124:PRINT nb → 124.

Un nom un peu plus long évite un REM. Ainsi, dans le programme ''CONSOMMATION'', nous n'avons pas légendé la variable ''DIST'', on comprend que c'est la ''distance''. Toutefois, des noms trop longs occupent de la mémoire, un octet par caractère.

Une seule obligation : tout nom de variable doit commencer par une lettre.

Si de nombreuses variables ont un lien commun, il est plus clair de les indicer. Par exemple pour des prix PRIX(0) à PRIX(10). Au-delà de 10, il faut déclarer un DIM. Exemple : DIM PRIX(24). S'il y en a six, vous avez intérêt à déclarer DIM PRIX(5) (= de 0 à 5) ; c'est une petite économie de mémoire, sinon le fait d'écrire seulement PRIX(1) = 215 provoque, à votre insu, un DIM PRIX(10), une réservation mémoire inutile.

LA COMMANDE DEFSTR

Vous voulez faire des économies de $ (au clavier...) ? Au début du programme, tapez, par exemple :

DEFSTR A-F

Cela implique que tous les noms de variables *commencant* par les lettres A, B, C, D, E et F correspondent à des chaînes. Vous pouvez alors écrire :

AMIE = ''Claudine'':B = ''PAUL''

Les étourderies du genre C$ = ''MARIE'' sont acceptées et vous obtiendrez les bonnes réponses en demandant indifféremment le nom avec ou sans $. Exemples : PRINT AMIE, PRINT AMIE$, PRINT C.

En revanche, FRIC = 124 provoquera un ''Type Mismatch Error'' (il manque les '').

LA COMMANDE DEFINT

Elle est aux nombres entiers ce que DEFSTR est aux chaînes. Les nombres entiers présentent trois avantages sur les nombres réels :
1 — Ils occupent deux octets au lieu de cinq.
2 — Les calculs et les boucles FOR-NEXT sont bien plus rapides.
3 — Ils s'arrondissent à l'entier le plus proche.
Exemple : I% = 27.628:PRINT I% → 28.
Un inconvénient : leur domaine va de + 32767 à − 32768.

Un bonne habitude consiste à mettre en début de programme DEFINT I − N, donc les variables commençant par I, J, K, L, M et N seront des entiers. Inutile de mentionner I%. C'est une norme héritée du langage FORTRAN, parce que I et N sont les premières lettres de ''INTEGER'' (= entier). C'est de là que vient cette habitude d'utiliser I, J et N comme variables de comptage du genre : FOR I = 1 TO 100 ou N = N + 1...

Et la vitesse ? Faites cet essai convaincant :

FOR I = 1 TO 10000:NEXT

Durée : 10,8 secondes. A présent, tapez DEFINT I − N et relancez cette même ligne. Durée : 5,7 secondes ! A l'occasion, faites ce même test sur d'autres micro-ordinateurs (même à plus de 10 000 FF),et vous verrez que l'AMSTRAD fait figure de ''Formule 1''.

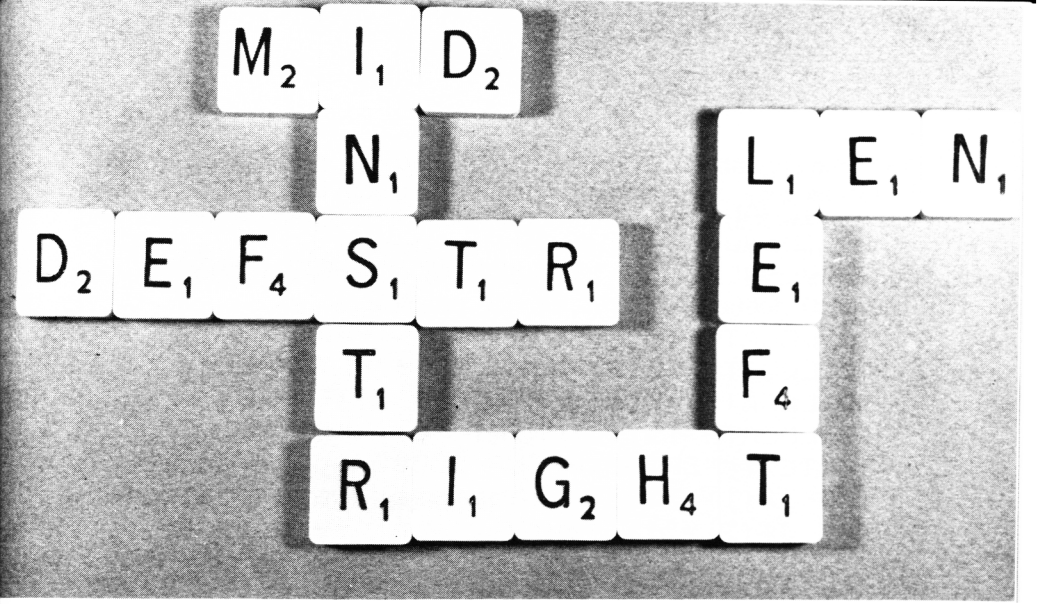
NOTE : Vous avez parfaitement le droit d'écrire DEFINT I − N, X − Z, H (entier de I à N, de X à Z et H).
Même remarque pour DEFSTR.

D'autre part, vous pouvez, plus loin dans le programme, modifier le DEFINT avec, par exemple, DEFREAL L − N,X.
Ainsi, les nouvelles valeurs de L, M, N et X seront des ''réels'' (décimaux et sans limites).

LA LISTE DES VARIABLES

Lorsque l'on commence à composer un programme qui s'annonce long et complexe, la sage précaution consiste à noter sur une feuille les différentes variables que l'on utilise, et la première ligne où elles apparaissent. En effet, deux pièges très classiques nous guettent :
1) Le changement d'orthographe. Exemple : on a calculé la valeur d'un total TOT = 628. Beaucoup plus loin, PRINT TOTAL → = 0.
2) Les noms déjà utilisés. Exemple : dans une boucle vous avez mis un compteur, le classique I = I + 1. Mais vous avez oublié que, dans un sous-programme, il y a une boucle d'attente de la forme FOR I = 1 TO 5000:NEXT. Donc, votre compteur débutera à 5001. Oh surprise !...

Cette liste rejoindra ensuite le listing et l'organigramme pour l'archivage. Elle sera super utile lors d'une modification ultérieure (la loi n° 2).



Chapitre VI

LES RESSOURCES DES FONCTIONS CHAINE

Les fonctions Basic destinées à traiter les chaînes sont beaucoup plus utilisées que les fonctions mathématiques ; même dans les programmes de calculs elles permettent de faire ''l'impensable''. Dans le manuel, vous avez sans doute parcouru ces diverses et nombreuses fonctions et leur mode d'action, mais en pensant souvent ''oui, et après ? A quoi va-t-elle me servir !'' Prises individuellement, à pas grand chose mais, en les combinant, on fait des merveilles, surtout pour décortiquer une réponse entrée par un INPUT.

Nous vous avons livré quelques menus échantillons dans le programme ''Consommation'' du chapitre IV en illustrant RIGHT$, UPPER$ et STRING$. Nous allons poursuivre par un module utilitaire qui sert dans presque tous les programmes et à plusieurs reprises. C'est la réponse à un menu quelconque, mais avec tous les raffinements possibles. Dans le listing qui suit, c'est le module débutant à la ligne 50000, ce qui précède n'est qu'un programme de démonstration.

```
1 ' CHAPITRE 6A
10 'MENU DE JEUX
20 CLS:LOCATE 11,3:PRINT "A QUOI JOUONS-
NOUS ?"
30 LOCATE 15,7:PRINT "B : BELOTTE"
40 LOCATE 15,9:PRINT "P : PETANQUE"
50 LOCATE 15,11:PRINT "T : TENNIS"
60 LOCATE 15,13:PRINT "G : mini-GOLF"
70 TEX$="BPTG":GOSUB 50000
100 J$(1)="Tu triches mieux que moi."
200 J$(2)="Il faut souvent se baisser...
"
300 J$(3)="Par une chaleur pareille !!!"

400 J$(4)="Tu joues beaucoup trop mal.."
1000 CLS:LOCATE 6,6:PRINT J$(K)
1010 LOCATE 13,12:PRINT "A : AUTRE JEU"
1020 LOCATE 13,14:PRINT "Q : on se QUITT
E"
1100 TEX$="AQ":GOSUB 50000
1110 ON K GOTO 1120,1130
1120 RUN
1130 CLS:LOCATE 4,12:PRINT "Je vais fair
e une bonne sieste..."
1140 LOCATE 1,22:PRINT ;
1200 END
50000 'REPONSE A UN MENU
50010 LT=LEN(TEX$):R$=""
50020 LOCATE 15-LT,24:PRINT "Reponse (";
50030 FOR I=1 TO LT-1
50040 PRINT MID$(TEX$,I,1);",";:NEXT
50050 PRINT RIGHT$(TEX$,1);")";CHR$(154)
;CHR$(243);CHR$(207)
50060 TEX$=UPPER$(TEX$)
50070 WHILE R$="":R$=INKEY$:WEND
50080 R$=UPPER$(R$):K=INSTR(TEX$,R$)
50090 IF K=0 THEN R$="":GOTO 50070
50100 RETURN
```

Le ''mode d'emploi'' est contenu dans la ligne 70. On désigne par TEX$ tous les symboles de réponses au menu qui précèdent, sans espace ni ponctuation. Au retour du module 50000, nous avons une valeur K qui est le numéro d'ordre. Exemple : Si la réponse était la lettre T, on a K = 3. La ligne 1000 donne une façon d'utiliser ce nombre K. Plus loin, nous proposons un autre menu (lignes 1010 à 1100). Au retour, nous utilisons la fontion ON... GOTO... avec K.

LE SOUS-PROGRAMME ''REPONSE AU MENU'' (utilitaire)

Analysons ligne par ligne :

50010 : LT est la largeur de la chaîne TEX$. On initialise R$ à vide.

50020 : Sur l'avant-dernière ligne d'écran, on va afficher le questionnaire Réponse. Sa tabulation horizontale est calculée en fonction de LT.

50030-50040 : Après la parenthèse ouverte en 50020, on va écrire chaque caractère de TEX$ suivi (ou espacé) par une virgule ; mais pas jusqu'au dernier qui, lui, sera suivi de la parenthèse fermée. Notez au passage que MID$ (TEX$,1,1) est l'équivalent de LEFT$(TEX$,1).

50050 : On termine par le dernier caractère, la parenthèse fermée, et trois caractères graphiques qui vous dessinent une jolie flèche désignant un carré en damier.

50060 : A présent, on décide que TEX$ est en caractères majuscules (une précaution).

50070 Voilà une ligne que vous aurez souvent à écrire. Notez-la dans votre pense-bête. La fonction INKEY$ lit le caractère du clavier qui vient d'être frappé ; on le baptise R$ (de Réponse). INKEY$ lit le clavier d'une manière *fugitive*, il n'arrête pas le déroulement du programme, il faut donc répéter sans cesse cette scrutation.
En ligne 50010, on a initialisé à vide R$, et la fonction WHILE... WEND revient à dire : ''tourne en rond si R$ est vide, mais R$ prend la valeur de INKEY$''. Donc, tant que l'on ne touche pas au clavier, le programme semble bloqué. En fait, il ''pédale'' sur cette ligne (il n'y a pas de curseur à l'écran comme avec INPUT).
Note : Cette ligne joue le rôle de la fonction GET (GET R$) présente sur certains micro-ordinateurs (APPLE, ORIC, etc.).

50080 : Un caractère venant d'être frappé, on le met en majuscule (une ''standardisation'' associée à la ligne 50060). Deuxième temps : on fait appel à la fonction INSTR (elle est très importante) qui va nous dire où se trouve R$ dans TEX$, sa place K.

50090 : Si R$ n'existe pas dans TEX$, K = 0. La réponse n'est pas correcte : on remet alors R$ à vide et on revient à la ligne 50070. Si non, RETURN.

Ce programme tapé et essayé, tapez :

```
DELETE -4000
```

Il ne reste alors que le module que vous allez enregistrer sur une cassette neuve : nom de cassette UTIL, nom du programme MENU. D'autres utilitaires iront le rejoindre sur cette précieuse cassette.

Les avantages de ce sous-programme sont très nombreux :

— On le recharge par la cassette UTIL sur un programme en cours d'écriture par :

```
MERGE ''MENU''
```

— Après chaque questionnaire-menu, plus besoin de taper de nombreuses lignes à base de IF. Simplement TEX$ = '' '':GOSUB 50000. C'est tout.
— Toutes les sécurités contre les reponses incorrectes sont incluses.
— Cette réponse monocaractère est dispensée d'une action sur Enter.
— TEX$ admet jusqu'à 13 caractères.

Voulez-vous un son BIP en cas de mauvaise réponse ? C'est très simple, modifiez ainsi la ligne 50090 :

50090 : IF K = 0 THEN R$ = '''':PRINT CHR$(7);:GOTO 50070 (n'oubliez pas le '';'' après le CHR$(7).

LES CHAINES DE BLANCS

Pour certaines tabulations ou pour effacer un mot ou une chaîne précise sur une page d'écran, on utilise une chaîne ne contenant que des blancs. Il est désuet de définir une chaîne de 20 espaces en faisant BL$ = '' suivi de 20 actions sur la barre d'espacement. Ecrivez tout simplement BL$ = SPACE$(20). Non seulement c'est plus clair et plus court (également en octets...), mais le nombre de blancs est paramétrable (paramétré = obtenu par un calcul), d'autant plus que ce nombre peut aller jusqu'à 255...

Ces chaînes de blancs sont indispensables pour effectuer des tabulations dans des programmes d'édition sur imprimante. Un exemple :

Votre imprimante fait 80 caractères par ligne, et vous devez centrer un titre TIT$ dont la longueur peut varier. La méthode infaillible est d'écrire :

```
BL$ = SPACE$((80 - LEN(TIT$))/2)
PRINT #8,BL$;TIT$
```

Et si cette formule aboutit à un SPACE$(5.8) ? Le Basic de l'AMSTRAD, vraiment très sympa, prend l'entier le plus proche, donc ici six blancs.

Ne pas confondre les fonctions SPACE$ et SPC (le manuel est très discret sur cette dernière...).

- SPC ne s'utilise que dans une expression PRINT, exemple PRINT SPC(12);TIT$.
- SPACE$ définit *aussi* une chaîne de caractères, ainsi BL$ = SPC(12) engendre un ''Syntax Error''.

LA CONCATENATION

C'est souder bout-à-bout plusieurs chaînes pour n'en faire qu'une. Exemple :

```
A$ = ''PAUL'':B$ = ''JEAN'':T$ = '' - ''
PRENOM$ = B$ + T$ + A$ (= JEAN-PAUL).
```

Vous obtiendrez le même résultat à l'écran avec ces trois écritures :

```
PRINT PRENOM$
PRINT B$;T$;A$
PRINT B$ + T$ + A$
```

Le troisième mode est encore une tolérance de l'AMSTRAD, car la plupart de ses concurrents renverraient un ''Syntax Error''.

LE CAS DE MID$

MID$ est à la fois une fonction (= qui renseigne) et une commande (= qui impose). Exemple avec A$ = ''ABCDE''
PRINT MID$(A$,3,2) donne ''CD'', mais on a aussi le droit d'écrire : MID$(A$,3,2) = ''XY'' (c'est une ''commande'').

En faisant ensuite PRINT A$, on a ''ABXYE''. Par contre, LEFT$ et RIGHT$ ne sont que des ''fonctions''.

Pour supprimer un caractère dans une chaîne, il ne faut pas écrire MID$(...) = '' '', ce serait sans effet. Reprenons A$ = ''ABCDE''.
MID$(A$,3,1) = CHR$(16) (CHR$(16) = DEL)
PRINT A$ donne ''ABDE'', mais attention, deux pièges : LEN(A1) est toujours égal à 5 ! Si on repasse une seconde fois cette même commande MID$, il n'y a aucun effet. Si on fait à présent MID$(A$,5,1) = CHR$(16) puis PRINT A$, on obtient ''ABD'' ! ET LEN(A$) est toujours égal à 5...

Donc, le ''caractère d'effacement'' CHR$(16) est à manier avec prudence.

En revanche, on a parfaitement le droit, par MID$, de substituer des caractères d'une chaîne par des blancs, par " " ou SPACE$, ou par tout autre caractère graphique.

Afin de vous montrer encore une astuce simple à l'aide de MID$, essayez ces quelques lignes :

```
1 ' CHAPITRE 6B
10 'MOT ARLEQUIN PAR MID$
20 MODE 0
30 A$="MOT ARLEQUIN"
40 PRINT:PRINT SPC(4);
50 FOR N=1 TO 12
60 PEN N:PRINT MID$(A$,N,1);:NEXT
70 PEN 1:PRINT:PRINT
```

NOTE : Sans PEN 1, en ligne 70, vous auriez une mauvaise surprise en revenant en Mode 1...

LES AUTRES FONCTIONS CHAINES

- STR$(123) transforme un nombre en chaîne, mais en lui mettant un blanc devant. Ainsi LEN(STR$(123)) = 4 et non pas 3.
- VAL est la fonction inverse de STR$. Si N$ = "123":VAL(N$) = 123. Quelques cas :

  N$ = "342FF":VAL(N$) = 342
  N$ = "A328":VAL(N$) = 0
  N$ = "34,56":VAL(N$) = 34
  N$ = " 3.12E3":VAL(N$) = 3120

- CHR$ inscrit le caractère dont le code ASCII est indiqué, mais de 32 à 255.

CHR$(0) à CHR$(31), à la suite de PRINT, sont des actions spéciales. Quelques exemples :

- 12 = CLS
- 16 = Effacement d'un caractère sous le curseur
- 17 = Effacement à gauche du curseur
- 18 = Effacement à droite du curseur
- 19 = Effacement du haut de l'écran
- 20 = Effacement du bas de l'écran
- 24 = Inversion PAPER/PEN
- 30 = Retour du curseur en LOCATE1,1

• ASC renvoie le code ASCII d'un caractère. Exemples : ASC(''A'') = 65 :ASC(''a'') = 97. (Il y a toujours une différence de 32 de majuscule à minuscule.)

• UPPER$, LOWER$ imposent les *lettres* en majuscules ou en minuscules. C'est la touche verte ''CAPS LOCK'' programmée. Il n'y a aucune action sur les signes graphiques ou de ponctuation. Du fait de leur présence (rare sur micro-ordinateurs), la fonction ASC est rarement utilisée sur l'AMSTRAD.

LE PRINT USING

Il permet d'écrire un nombre selon un format bien établi en le représentant *comme une chaîne* où le point décimal occupe une position fixe : Exemples avec PRINT USING ''# # # . # #'';N

N = 123.4567 : 123.46 (il arrondit)
N = 12 : 12.00
N = .1234 : 0.12
N = 0 : 0.00

Les nombres sont bien en colonnes et non plus imprimés à partir de la gauche.

Si N sort du format établi, exemple N = 1234.56, ce bel alignement est détruit, mais N est affiché sous la forme %1234.56. Le signe % indique qu'il y a eu dépassement.

Attention au signe − qui occupe un caractère. Ainsi il y aura dépassement avec N = − 123. Pour avoir toujours le signe + ou − devant le nombre, faire le format :

PRINT USING ''+ # # # . # #'';N

Pour un programme d'édition sur imprimante, la syntaxe est la suivante :

PRINT #8,USING''# # # . #''.

NOTE : S'il s'agit de nombres à représenter sans décimales, le format est bien sûr ''# # # #'', avec toujours le chiffre de droite arrondi au plus proche.

Le PRINT USING appliqué aux chaînes présente beaucoup moins d'intérêt. Citons le format PRINT USING'' & '';A$. Le signe & (ET commercial ou AMPERSAND) représente A$ en entier, lequel sera entouré du même nombre de blancs, à gauche et droite, que &.

Au chapitre I, nous avons utilisé le PRINT USING sur imprimante pour le tableau ''Réglette de Tabulation'' avec le format ''# #''. En voici le programme (uniquement à titre documentaire) :

```
1 ' CHAPITRE 6C
10 ' REGLETTE DE TABULATION
20 PRINT#8,SPC(2);"REGLETTE DE TABULATIO
N"
30 PRINT#8,STRING$(26,"_")
40 PRINT#8
50 PRINT #8,"MODE 0";SPC(4);"MODE 1";SPC
(4);"MODE 2"
60 PRINT #8
70 FOR I=2 TO 20 STEP 2
80 PRINT #8,USING"##";I;:PRINT#8,SPC(2);
:PRINT #8,USING"##";(20-I)/2+1;:PRINT#8,
SPC(4);
90 PRINT #8,USING"##";I;:PRINT#8,SPC(2);
:PRINT #8,USING"##";(40-I)/2+1;:PRINT#8,
SPC(4);
100 PRINT #8,USING"##";I;:PRINT#8,SPC(2)
;:PRINT #8,USING"##";(80-I)/2+1;:PRINT#8
,SPC(4);CHR$(13)
110 NEXT
120 FOR I=22 TO 40 STEP 2
130 PRINT#8,SPC(10);
140 PRINT #8,USING"##";I;:PRINT#8,SPC(2)
;:PRINT #8,USING"##";(40-I)/2+1;:PRINT#8
,SPC(4);
150 PRINT #8,USING"##";I;:PRINT#8,SPC(2)
;:PRINT #8,USING"##";(80-I)/2+1;:PRINT#8
,SPC(4);CHR$(13)
160 NEXT
170 FOR I=42 TO 60 STEP 2
180 PRINT#8,SPC(20);
190 PRINT #8,USING"##";I;:PRINT#8,SPC(2)
;:PRINT #8,USING"##";(80-I)/2+1;:PRINT#8
,SPC(4);CHR$(13)
200 NEXT
300 '_____ fin de listing _____
```

Nous avons fait le tour de l'arsenal pour démantibuler les chaînes ou pour les créer. Il ne reste guère que HEX$ qui donne la représentation en hexadécimal (hexadéci = seize) d'un nombre inférieur à 256, et BIN$, la représentation binaire d'un nombre inférieur à 65535. Essayez PRINT BIN$(64333).

Chapitre VII

MANIPULONS LES FONCTIONS NUMÉRIQUES

Rassurez-vous, l'auteur n'est vraiment pas passionné par les mathématiques, il s'en sert lorsqu'elles résolvent ses problèmes ; nuance.

Le Basic de l'AMSTRAD est très copieux en ce domaine et de ce fait l'écriture s'en trouve très simplifiée. Si vous avez déjà une bonne pratique du Basic sur d'autres micro-ordinateurs, vous pourrez ranger dans les oubliettes toutes ces longues et nombreuses formules à base de ''INT'', car ici ROUND, FIX et MOD se chargeront de cela.

Pour beaucoup d'entre vous, certaines fonctions mathématiques telles que SIN, COS, etc... sont du ''chinois'' ou quelques vagues souvenirs de jeunesse. Il n'empêche que vous les recopiez d'un livre ou d'une revue afin d'obtenir de superbes figures, rosaces, cercles, etc. Aussi, afin de ne pas ''taper idiot'', nous vous dirons à quoi elles correspondent, et vous pourrez alors les triturer vous-même. Et il y a de belles et curieuses choses à faire...

Pour plus de clarté, nous avons classé ces fonctions par genres.

LES FONCTIONS OPERATEURS

Vous connaissez +, −, * et /, mais à cela s'ajoute MOD : très précieux !

C'est l'abréviation de ''MODULO'' ; le module est le RESTE d'une division. Quand on divise 13 par 5, on obtient 2 et un reste de 3. D'accord ? Alors essayons :

PRINT 13 MOD 5 et l'écran affiche 3.

Cela s'écrit comme une division (13/5), mais en remplaçant le signe ''/'' par MOD. MOD ne s'applique qu'aux nombres entiers.

La fonction FIX renvoie la partie *non décimale* d'un nombre, ainsi :

PRINT FIX(−34.728) donne −34

alors que INT(−34.728) donnerait −35...

MOD et FIX sont souvent combinés. Ainsi, un chronomètre a donné un temps TM = 214 minutes. A quoi cela correspond-il ?

Heures = FIX(214/60)

plus 214 MOD 60 minutes.
Ce qui donne 3 heures et 34 minutes. Avouez que ces deux écritures sont à la fois courtes et simples.

Reprenons cela d'une manière plus concrète avec ce petit programme qui décompose un temps en secondes en heures, minutes et secondes.

```
200 ' FIX & MOD
210 SPH= 3600:'secondes par heure
220 SPM=   60:'secondes par minute
230 INPUT"  NOMBRE DE SECONDES: ",NS
240 IF NS>32767 THEN PRINT "SUPERIEUR A
32767":GOTO 230
250 H=FIX(NS/SPH):R=NS MOD SPH
260 M=FIX( R/SPM):S= R MOD SPM
270 PRINT:PRINT "=";H;"heures,";M;"minut
es et";S;"secondes"
280 PRINT STRING$(40,"-"):GOTO 230
290 END
```

La ligne 240 est importante car les opérations MOD n'acceptent que des valeurs d'entrées ''entières'', c'est-à-dire comprises entre −32768 et +32767. Si on faisait 7000 MOD SPH, il y aurait le message d'erreur ''OVERFLOW''. En revanche, 37.67 MODE 10 donne 8. Il a arrondi 7.67 à l'entier le plus proche.

Vous remarquerez que les calculs pour cette conversion résident dans les deux lignes 250 et 260, et ces lignes sont courtes.

Vous remarquerez que les calcules pour cette conversion résident dans les deux lignes 250 et 260, et ces lignes sont courtes.

NOTE : Le signe " \ " (anti-slash) est peu utile car A \ B équivaut à FIX(A/B), mais limité aux nombres entiers.

LA FONCTION ROUND

C'est la super merveille ! En effet, la moindre division va vous donner neuf chiffres, et une telle précision est très souvent absurde sur le plan pratique, et très encombrante sur l'écran ou sur l'imprimante. Deux chiffres après la virgule vous conviennent ? Alors écrivez :

PRINT ROUND(113/31,2)

donne 3.56, alors que 113/31 = 3.64516129. Vous remarquerez que ROUND *arrondit* au plus proche le chiffre de droite.

Cela vous rappelle le calibrage par PRINT USING. Certes, l'effet après un PRINT est pratiquement le même. Mais attention ! PRINT USING transforme un nombre en chaîne, ROUND transforme un nombre en un autre. Vous avez le droit d'écrire :

N = ROUND(A/B,1) + 5

chose impossible avec PRINT USING.

ROUND est le remède miracle à certaines farces communes à TOUS les micro-ordinateurs, par exemple celle d'une boucle FOR... NEXT avec un STEP décimal.

Essayez ces quelques lignes anodines et attendez-vous à une surprise :

```
10 ' ROUND ET STEP FRACTIONNAIRES
20 '-----STEP POSITIF-----------
30 CLS
40 FOR I=0 TO 1 STEP 0.05
50 PRINT ROUND(I,2),I
60 NEXT:PRINT TAB(6);I
70 END
```

En comptant en binaire, même sur cinq octets, il y a toujours un "bit non satisfait" et, à force de les cumuler, cela finit par se voir. Pire encore, la valeur I = 1 n'a pu être obtenue. Sur la colonne de gauche à l'écran, on constate l'effet bienfaisant de la fonction ROUND. C'est encore plus spectaculaire avec un STEP négatif.

```
100 '------STEP NEGATIF----------
110 CLS
120 FOR I=1 TO 0 STEP -0.05
130 PRINT ROUND(I,2),I
140 NEXT:PRINT TAB(6);I
150 END
```

Et voici enfin l'arme absolue : ajoutez la ligne 45 (et idem en 125).

45 I = ROUND(I,2)

On finit bien à 1 et à 0.

NOTE : En fin de boucle FOR... NEXT, nous faisons apparaître la valeur finale de I qui est égale à la valeur cible + le step (rappel important commun à tous les Basic).

LES RACINES

Les racines de nombres sont ici banales : on a SQR pour les racines carrées.

SQR(81) = 9 (ou 81↑0.5 = 9)

Pour les racines cubiques, on utilise les exposants fractionnaires :

27↑(1/3) = 3

N'oubliez pas les parenthèses avec le symbole "d'exponentiation" "↑" car il est plus prioritaire que celui de la division "/". Ainsi, 27↑1/3 fera d'abord 27↑1 = 27, et division par 3. Résultat : 9...

LES FONCTIONS DE "NATURE"

Ce sont celles qui modifient la nature d'un nombre. Les deux plus importantes sont ABS, INT et SGN, mais bien après ROUND dont on a déjà parlé longuement.

ABS renvoie la valeur "absolue" d'un nombre, c'est-à-dire sans signe, donc positif. Plus simplement, disons qu'elle supprime le signe moins s'il y en a un.

ABS(− 34.78) = 34.78

La fonction INT arrondit à *l'entier le plus faible.*

INT(34.78) = 34, mais INT(− 34.78) = − 35

parce que − 35 est *plus petit* que − 34.

A noter que la fonction INT devient moins fréquente sur AMSTRAD que sur d'autres Basic en raison de la présence de ROUND et FIX.

La fonction SGN (de ''signus'') indique le signe d'un nombre, il renvoie 1, 0 ou −1.

```
SGN(612.14) = 1:SGN(-28) = -1;SGN(0) = 0
```

Cette fonction est souvent très pratique par sa brièveté. Imaginez le nombre de IF qu'il faudrait pour dire : ''si les nombres A et B ne sont pas de même signe, mettre C à zéro''. Cela devient :

```
IF SGN(A) + SGN(B) = 0 THEN C = 0
```

Un autre exemple : il faut que A soit amené au même signe que C. C'est tout simplement :

```
A = ABS(A) * SGN(C)
```

Viennent ensuite les fonctions de conversions d'entiers en réels et vice-versa (usage peu fréquent).

La fonction CINT (=conversion in integer) arrondit à l'entier le plus proche (contrairement à INT), et l'encombrement mémoire passe de 5 à 2 octets. Ainsi, CINT(34.78) = 35..

La fonction inverse est CREAL (=convert in real).

La fonction UNT est d'un emploi super rarissime, mais expliquons-la.

En Basic, il y a deux sortes de nombres qui tiennent sur deux octets, les ''entiers'' (genre I%) de −32768 à +32767, et les numéros de lignes et les adresses mémoire de 0 à 65535. Tous sont non-décimaux, mais leurs écritures en binaire sont différentes.

De 0 à 65535, on utilise les 16 bits, soit 2↑16 = 65536 combinaisons.

Pour les ''entiers'', 1 bit est consacré au signe + ou −, il reste donc 15 bits. Or, 2↑15 = 32768 combinaisons (zéro compris) (bit de gauche = 1 = négatif).

UNT(N) va traduire l'image binaire de N en ''entier''. Si N est inférieur à 32768, il n'y a pas de changement.

N% = UNT(32000) donne N% = 3200, mais N% = UNT(40000) donne N% = −25536, parce que le bit de gauche (dit ''le plus significatif'' est 1 et non pas zéro). Faisons PRINT BIN$(40000) et PRINT BIN$(−25536), nous obtenons exactement la même image binaire dans les deux cas.

LA FONCTION DEF FN

Lorsqu'une formule de calcul est utilisée de nombreuses fois dans un programme, on a deux solutions pour ne pas la réécrire chaque fois : un GOSUB vers un sous-programme d'une ligne, ou créer une fonction mathématique ''maison'' par DEF FN, qui signifie Définition de Fonction. On définit cette fonction une bonne fois pour toutes vers le début

du programme et on l'appelle autant de fois qu'il est nécessaire. Un exemple super simple :

On dispose déjà de SQR(A) qui fournit la racine carrée d'un nombre A ; on veut une fonction maison qui donne la racine cubique, on l'appelera par FN RACUB(A). ''RACUB'', c'est un nom à nous, FN sert à prévenir le Basic que c'est une fonction définie plus haut dans le programme.

20 DEF FN RACUB(N) = N↑(1/3)

En fait, on réserve les DEF FN pour des formules plus longues à écrire, comme celle de la ligne 50 du programme suivant où nous profitons de l'occasion pour présenter des résultats formatés par ROUND (lignes 80 et 100), et comment FIX déjoue le plus célèbre des problèmes pièges du ''Certificat d'Etudes'' (ligne 90).

```
10 ' DEF FN pour les formules complexes
20 ' Diametre d'une sphere de fer.(densi
te du fer=7.8)
30 ' volume de la sphere=PI*R^3*(4/3)
40 ' poids P=volume * densite
50 DEF FN DIAMETRE(P)=((P/(7.8*4/3)/PI)^
(1/3))*2
60 CLS
70 INPUT"  Poids en GRAMMES d'une sphere
: ",P
80 PRINT "  DIAMETRE=";ROUND(FN DIAMETRE
(P),2);"cm"
90 Q=(FIX(100/FN DIAMETRE(P)))^3:' Q=Qua
ntite de spheres dans un container de un
 metre de cote.
100 PT=ROUND(P*Q/1000000,3)
110 PRINT "  soit";Q;"dans un metre cube
"
120 PRINT "  ce qui fait un poids de";PT
;"tonnes"
130 PRINT STRING$(40,"-"):GOTO 70
140 END
```

```
RESULTAT:

 Poids en GRAMMES d'une sphere: 100
 DIAMETRE= 2.9 cm
 soit 39304 dans un metre cube
 ce qui fait un poids de 3.93 tonnes
-------------------------------------
 Poids en GRAMMES d'une sphere: 7512
 DIAMETRE= 12.25 cm
 soit 512 dans un metre cube
 ce qui fait un poids de 3.846 tonnes
-------------------------------------
 Poids en GRAMMES d'une sphere: 150000
 DIAMETRE= 33.24 cm
 soit 27 dans un metre cube
 ce qui fait un poids de 4.05 tonnes
-------------------------------------
 Poids en GRAMMES d'une sphere: 160000
 DIAMETRE= 33.96 cm
 soit 8 dans un metre cube
 ce qui fait un poids de 1.28 tonnes
-------------------------------------
```

On aurait bien tort de croire que le DEF FN soit limité à une seule variable, nous le prouvons avec celui-ci à trois paramètres : c'est un ''remake'' très dépouillé de notre précédent ''Consommation de carburant''.

```
200 ' DEF FN a plusieurs variables
210 ' Consommation; version simplifiee
220 DEF FN CONSOM(KA,KD,L)=ROUND(100*L/(
KA-KD),1)
230 CLS
240 INPUT "KM depart:",KD
250 INPUT "KM arrivee:",KA
260 INPUT "litres de plein a plein:",L
270 PRINT "ce qui fait";FN CONSOM(KA,KD,
L);"litres aux 100 km"
280 PRINT STRING$(40,CHR$(210)):GOTO 240
290 END
```

En ligne 220, vous constatez que l'ordre des variables annoncées peut être différent de celui dans lequel on les utilise. En revanche, en ligne 270, il faut les mettre dans les parenthèses dans l'ordre pré-établi.

NOTE : Le CHR$(210), dans le STRING$ de la ligne 280, dessine à l'écran un trait horizontal continu et épais.

RAPPEL : Un DEF FN ne peut être défini que dans une ligne de programme, pas en mode direct. Mais le programme achevé, on peut *utiliser* cette fonction FN en mode direct.

LES FONCTIONS D'ANGLES OU TRIGONOMETRIQUES

PI, DEG, RAD, SIN, COS, TAN, ATN.

Comme nous l'avions annoncé, ce paragraphe est destiné à ceux et à celles qui ignorent la trigonométrie ou qui l'ont oubliée. Nous ne dirons que ce qui est (très) utile de savoir en programmation.

En examinant la figure VII 1, considérons un angle ''A'' dans un cercle, une sorte de part dans un camembert. Ce cercle a un rayon égal à 1, c'est important ; l'unité est sans importance. Appelons ''O' le centre de cercle et ''P'' le point où cet angle ''coupe'' le cercle.

Imaginons que le point P soit dans un graphique ordinaire avec ses coordonnées en x et y. L'ABSCISSE x porte le nom de COSINUS (COS), et l'ORDONNEE y s'appelle SINUS (SIN).

Imaginons à présent que cet angle A va varier de 0 à 90°.

Quand A = 0 COS(A) = 1 et SIN(A) = 0
Quand A = 90° COS(A) = 0 et SIN(A) = 1

A présent, l'angle continue d'augmenter, 120° par exemple. Le SINUS va décroître, mais le segment COSINUS passe de l'autre côté du point O. D'accord ? Il prend alors une valeur *négative.* Quand l'angle atteint 180°, on a SIN(A) de nouveau à 0, mais COS(A) = − 1.

L'angle A continue sa croissance de 180 à 270° (3/4 de cercle) : COS(A) va passer progressivement de − 1 à 0, tandis que c'est au tour du segment SINUS de passer de l'autre côté du point O, et il devient négatif jusqu'à − 1.

De 270° à 360°, le SINUS va passer de − 1 à 0 et le COSINUS va croître de 0 à + 1. 360° ou 0°, c'est pareil.

En somme, SINUS et COSINUS sont les coordonnées des points du cercle ; voilà pourquoi on les utilise pour dessiner un cercle à l'écran.

Le court programme qui suit va illustrer notre exposé, mais nous avons pris les précautions suivantes :

a) Le point de coordonnée 0, 0 sur l'AMSTRAD se trouve dans l'angle inférieur gauche, on va le mettre au centre de l'écran par la fonction ORIGIN (ligne 320).

b) Pour que le cercle occupe toute la hauteur de l'écran, on va lui donner un rayon de 199 ; donc les COS et SIN seront multipliés par 199.
c) Pour que le tracé soit plus rapide, on va demander un point du cercle tous les deux degrés. En ligne 350, on pourrait mettre un STEP de 0.5 afin d'avoir un superbe tracé continu, mais ce serait quatre fois plus long.
d) En fin de tracé (ligne 380), nous remettons les choses comme nous les avons trouvées, c'est-à-dire ORIGIN 0,0 (c'est super prudent...), et RAD (=radian) comme unité d'angle par défaut. Nous expliquerons plus loin ce qu'est le radian.
e) En ligne 330, ce point va nous indiquer le centre (facultatif).

RAPPEL : Avec ORIGIN au milieu de l'écran, les points situés dans le quart gauche inférieur ont des coordonnées x, y négatives.

```
300 ' CERCLE AVEC ORIGINE AU CENTRE
310 CLS:BORDER 9
320 ORIGIN 320,200
330 PLOT 0,0,1
340 DEG
350 FOR A=0 TO 360 STEP 2
360 PLOT COS(A)*199,SIN(A)*199
370 NEXT
380 ORIGIN 0,0:RAD
390 END
```

Pour dessiner une élipse, modifiez la ligne 360 pour que SIN(A) ne soit multiplié que par 100 au lieu de 199.

LES RADIANS

On connaît le nombre PI = 3,14... pour calculer la circonférence ou la surface d'un cercle, mais on dit aussi que 360° vaut deux PI radians, donc le radian fait 57,2957... degrés. Le cercle complet fait deux PI. Dans le programme précédent, supprimons la ligne 340 ; l'unité d'angle est alors le radian, et la ligne 350 devrait donc s'écrire :

```
350 FOR A=0 TO 2*PI STEP PI/90
```

En effet, PI = 180°, donc PI/90 = 2°.

Qu'avons-nous gagné ? Absolument rien ! Que ce soit en degrés ou en radians, les durées d'exécution sont toutes deux égales à 6,7 secondes.

Il faut dire aussi que la fonction DEG est super rarissime sur les autres micro-ordinateurs, d'où une certaine ''habitude'' chez ceux qui ont rédigé les petits programmes du manuel d'origine... Donc, si vous préférez raisonner en degrés, ne vous gênez surtout pas !

LA FONCTION TAN

Elle signifie ''tangente'' (figure VII 1) et elle est peu utile pour faire des arabesques sur l'écran, par contre elle est très pratique pour calculer le côté d'un triangle rectangle dont on connaît l'autre côté et l'angle opposé. Attention ! Pour A = 90° TAN(A) = infini d'où ''OVERFLOW''.

La fonction ATN (= arc tangente) fait l'opération inverse. Exemple (en degrés) : TAN(30) = 0.57735 ; ATN(0.57735) = 30°.

LES FONCTIONS DU HASARD
RND ET RANDOMIZE

Si vous faites PRINT RND, vous obtenez un nombre inférieur à 1, mais supérieur ou égal à zéro, et ce avec neuf chiffres après le point décimal.

Si vous désirez un nombre aléatoire compris entre 0 et 100 et non décimal, il suffit d'écrire :

```
PRINT ROUND(RND*100)
```

Si vous faisiez INT(RND* 100), la valeur serait comprise entre 0 et 99 parce que INT(99.99) = 99. Exemple pour composer votre LOTO :

```
10 FOR N=1 TO 7
20 PRINT ROUND(RND*48)+1;
30 NEXT
```

Expliquons la ligne 20. Il faut des nombres compris entre 1 et 49, le zéro est interdit. Il suffit de programmer le hasard de 0 à 48, d'y ajouter 1, et le tour est joué (ce serait merveilleux si ces trois petites lignes vous faisaient rembourser le prix de ce livre...).

Ces nombres aléatoires décimaux et inférieurs à 1 sont écrits à demeure dans la mémoire ROM, et ce dans un ordre immuable. Faites :

```
FOR I=1 TO 3:PRINT RND:NEXT
```

vous obtenez :

```
0.271940568
0.528611238
0.021330127
```

Ce sont les trois premiers. En relançant ce programme, vous aurez les trois *suivants,* etc. Faites un ''RESET'' par SHIFT = CTRL + ESC et relancez cette ligne : encore ces trois nombres 0.27194... etc. Conclusion : les dés sont pipés.

La fonction RANDOMIZE peut améliorer ou fausser encore ce hasard. Elle fixe le ''départ'' dans la (très longue) liste des nombres aléatoires. A la mise sous tension, c'est RANDOMIZE(0).

Si l'on commande RANDOMIZE(90), le premier PRINT RND donne *toujours* 0.209440658, le suivant 0.7786112386, etc.

Essayez ceci *après* un RESET :

```
10 FOR I=O TO 20:RANDOMIZE(I)
20 PRINT I,RND;RND:NEXT
```

Vous constaterez que le premier changement intervient avec I = 5. En relançant par RUN, vous obtenez exactement le même tableau !

Alors, comment faire du VRAI HASARD ? En faisant RANDOMIZE(TIME). TIME est un nombre réel qui indique le nombre de trois-centièmes de seconde qui sont écoulés depuis la mise sous tension (ou le dernier RESET).

NOTE : Dans notre programme LOTO, il y a des centaines de valeurs de RND qui peuvent donner ''34'' puisque c'est un arrondi.

On améliorera sa ''fiabilité hasardeuse'' en ajoutant la ligne 5 :

```
5 RANDOMIZE(TIME)
```

AUTRES FONCTIONS NUMERIQUES

MAX et MIN permettent de ''sortir'' le champion ou le tocard dans une série de nombres ou de variables mise entre parenthèses. Elles ne peuvent s'appliquer à un tableau DIM NB(8), genre PRINT MAX(NB(8)).

Nous ne parlerons pas des fonctions logarithmiques (ou exponenetielles) telles que LOG, LOG10 et EXP. Elles sont utiles pour certains calculs scientifiques, de physique en particulier, mais nous ne voyons pas l'utilité de faire un mini-cours de maths à l'intention de ceux qui les ignorent, et qui peuvent continuer ainsi à vivre et programmer heureux. Ce qui n'était pas le cas pour les fonctions trigonométriques (COS, SIN), qui elles sont d'usage très courant en graphisme.

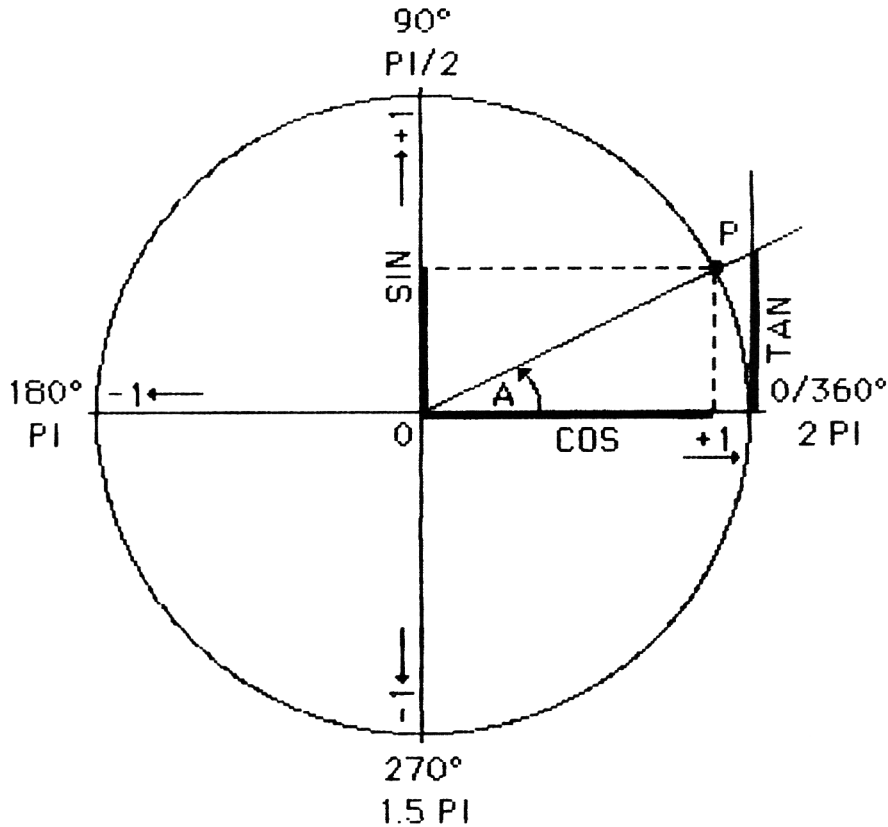


Figure VII 1
Représentation graphique des fonctions trigonométriques SIN, COS et TAN. Le cercle a un rayon égal à 1.

Chapitre VIII

LES COULEURS

La programmation des couleurs sur l'AMSTRAD est plutôt ''indirecte'', pas compliquée, mais pas simple non plus. Que les surdoués qui ont tout compris dès la première lecture du manuel d'origine se fassent connaître !...

Pour tous ceux qui avaient renoncé à comprendre — ils sont nombreux et excusables — nous allons tout d'abord présenter les faits d'une façon toute nouvelle.

Il y a CINQ choses différentes :

1) Le numéro de référence de la teinte, que nous appelons RT, c'est 0 = noir jusqu'à 26 = blanc brillant.
2) La palette de teintes que l'on se sélectionne, c'est INK. En mode 1, mode normal, on n'a droit qu'à quatre INK, numérotés de 0 à 3. Dès la mise sous tension, les INK ''par défaut'' sont les suivants :

INK 0,1 = bleu (RT = 1 = bleu)
INK 1,24 = jaune vif (RT = 24 = jaune vif)
INK 2,20 = turquoise vif
INK 3,6 = rouge vif

La turquoise ne vous intéresse pas ? Vous aimeriez du vert (RT = 9), faites simplement :

INK 2,9 (en modes direct ou programmé)

En mode 0, la palette va de INK 0 à INK 15, par défaut les quatre premiers sont les mêmes que ci-dessus, et voici la suite :

INK 4,26 = blanc brillant
INK 5,0 = noir
INK 6,2 = bleu vif
etc... jusqu'à
INK 15,16,11 = rose et bleu-ciel alternés (beurk !).

3) PAPER, c'est la couleur du fond. Par défaut, il a la couleur de INK 0, mais si cela ne vous convient pas, faites :

PAPER 3

et il prend ainsi la couleur de l'INK 3, à savoir ici rouge vif. Si vous faites PAPER 3,2 vous aurez alternativement les INK 3 et 2 et comme on avait modifié INK 2 en vert, attention les yeux...

4) PEN (= stylo en anglais), c'est la couleur des caractères, toujours définie comme PAPER avec le numéro de l'INK. Par défaut, c'est INK 1. Voilà pourquoi, à la mise sous tension (options par défaut), on a un fond bleu avec lettres jaune vif. On peut bien sûr changer cette couleur en faisant PEN 2, par exemple.
5) BORDER, c'est la couleur de la bordure, elle est *totalement indépendante,* tellement qu'elle ne se définit pas avec un numéro d'INK, mais avec un RT. Exemple : BORDER 21 = jaune citron.

QUELQUES REMARQUES

1)Si PAPER et PEN ont le même numéro d'INK, c'est pire qu'un CLS... Cette étourderie peut arriver, et l'on croit que l'ordinateur est ''planté'' car le clavier ''ne répond plus''. Faites ESC puis PEN 1 (ou 2 ou 3) et ENTER.
2) En Mode 1, faisons PEN 7, il n'y a pas de signal d'erreur et on obtient des lettres rouges. Pourquoi ? L'ordinateur prend la suite après INK 3 en repartant de 0, ainsi la série 4, 5, 6, 7, 8... se traduit par 0, 1, 2, 3, 0... Donc, faire PEN 7 équivaut à PEN 3 et comme INK 3 = rouge vif... De même, en MODE 2, il renumérote 0, 1, 0, 1, etc.
3) Pour changer les couleurs de l'écran, i suffit de modifier les INK 0 et 1 puisque *par défaut* le fond sera en INK 0 et les caractères en INK 1. C'est ce que nous avons fait dans notre programme ''KEY'' du chapitre I afin d'obtenir lettres noires sur fond gris par la touche ''4'' (c'est la combinaison qui donne la meilleure netteté).
4) Vous avez pu être désagréablement surpris que, outre le manque de simplicité, on ne dispose, en mode normal, c'est-à-dire en 40 caractères par ligne, que de quatre teintes simultanées, alors que des micro-ordinateurs concurrents en ont 7 ou 16. C'est parce que l'AMSTRAD propose 20, 40 ou 80 caractères par ligne sur seulement 16 kilo-octets de mémoire, et ce avec graphisme mixable au texte. Cela a été résolu par une gestion très spéciale (et très complexe) de la mémoire d'écran qui empêche plus de 4 couleurs en MODE 1 et plus de 2 couleurs en MODE 2. On peut se consoler par les possibilités en MODE 2 : traitement de texte, tableurs (''CALC'')

et graphisme haute définition. Par rapport à ses concurrents, l'AMSTRAD est donc un peu moins ludique et bien davantage professionnel. A titre d'exemple, une "carte 80 colonnes" pour APPLE II coûte une petite fortune, ici c'est gratuit et sans ouverture du capot, alors ne nous plaignons pas... Si vous voulez faire une "débauche de couleurs", passez en MODE 0, mais en évitant d'être bavard, car les caractères sont vraiment énormes.

LE CHOIX DES COULEURS

Il faut savoir combiner des considérations purement artistiques et d'autres purement techniques : la juxtaposition de certaines couleurs est insoutenable pour les yeux ; dans d'autres cas, les couleurs semblent baver l'une sur l'autre, et le texte est alors très difficile à déchiffrer. N'accusez pas le moniteur couleur car vous auriez exactement le même effet sur n'importe quel téléviseur couleur. Le tube qui équipe le moniteur AMSTRAD est rigoureusement identique à ceux qui équipent les téléviseurs couleur (1) et tout tient dans leur principe de la restitution des couleurs.

LA VIDEO COULEUR

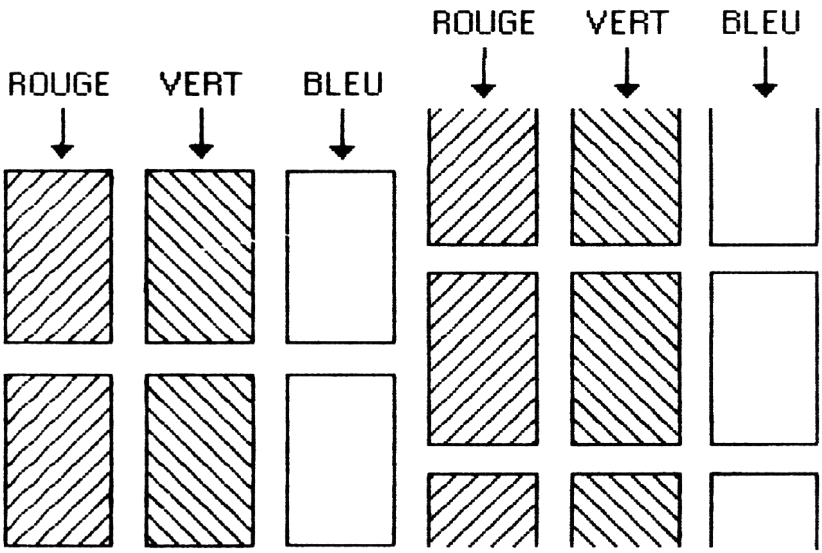
A l'aide d'une loupe, examinez la surface d'un écran couleur représentant du blanc : on ne voit que des petits rectangles rouges, verts et bleus également lumineux. Vue de loin, notre rétine fait une "synthèse additive" et ne voit que du blanc. Le blanc n'est pas une couleur, mais le mélange équitable de toutes les couleurs. Le noir est *l'absence* de toute couleur. Cette disposition en petits rectangles est illustrée sur la figure VIII 1.

Faites un écran rouge et examinez-le à la loupe. On observe une ligne verticale de rectangles rouges suivie de deux lignes verticales *noires,* c'est normal puisque les rectangles "photophores" bleus et verts sont éteints. C'est en combinant ces trois "couleurs primaires" tout en dosant leurs intensités respectives que l'on peut obtenir 27 teintes. D'autre part, remarquez ce décalage vertical des rectangles d'une verticale à l'autre. Voilà pourquoi il y a des couleurs qui bavent horizontalement et d'autres verticalement ; pourquoi certaines couleurs de lettres amènent une sorte de liseré noir (ou autre) sur le côté.

Il y a 702 combinaisons ($27 \times 26 = 702$) de couleurs PAPER/PEN, voilà qui vous laisse le champ libre à bien des essais. L'auteur ne les a pas toutes testées (vous vous en doutiez...), mais en MODE 2, la combinaison qui rend la meilleure netteté est INK 0,13 ; INK 1,0. Pour tester la netteté, faites une ligne de "w" minuscules...

Une autre association nette et reposante pour la vue est INK 0,1 ; INK 1,16. Elles figurent dans notre programme "KEY". Vous pourrez certainement en découvrir de meilleures.

(1) A l'exception du tube TRINITRON de SONY.



R + V + B	Blanc
R + B	Magenta (rouge violacé)
V + B	Cyan (bleu-vert)
R + V	Jaune

Figure VIII 1
La surface d'un écran TV couleur vue à la loupe.

SEGMENTATION DES COULEURS

Fort heureusement, il est très facile de changer plusieurs fois les couleurs de PAPER et PEN sur une *même ligne* de texte. Le petit programme qui suit n'a aucune prétention artistique, son rôle est de vous montrer quelques ''acrobaties'' simples (en MODE 1) :

```
10 '    COULEURS SEGMENTEES
20 CLS
30 LOCATE 17,11:PAPER 3:PRINT " PAPER 3
";:PAPER 0
40 LOCATE 17,10:PAPER 2:PRINT SPC(9);:PA
PER 0
50 LOCATE 17,12:PAPER 2:PRINT SPC(9);:PA
PER 0
60 LOCATE 3,17:FOR I=1 TO 3
70 PAPER I:PRINT SPC(12);
80 NEXT:PAPER 0
90 LOCATE 1,20:PRINT "PEN:";
100 FOR I=0 TO 15
110 PEN I:PRINT I;:NEXT
120 PEN 1
130 A$="MOSAIQUE"
140 LOCATE 18,4:FOR I=1 TO LEN(A$):C$=MI
D$(A$,I,1)
150 PAPER I-1:PEN I:PRINT C$;
160 NEXT:PAPER 0:PEN 1
170 END
```

1) Ligne 30 : A la suite d'un LOCATE, PAPER passe en rouge pour la chaîne ''PAPER 3'' ; puis on revient en PAPER bleu. On a donc le texte écrit sur une petite bande rouge.
2) Lignes 40 et 50 : Au-dessus et au-dessous de ce ''bandeau'' rouge, on accolle deux segments bleu clair ; même méthode, mais la chaîne a été remplacée par neuf blancs, SPC(9).
3) Lignes 60 à 80 : Cette fois, trois bandes de PAPER différents sont mises bout-à-bout. La technique est toujours la même.
4) Lignes 90 à 120 : Nous illustrons ce qui a été dit plus haut au sujet de la ''rotation'' des PEN de 0 à 3 sur un cycle de 0 à 15.
5) Lignes 130 à 160 : On combine les changements de PAPER et de PEN ; ainsi, pour chaque caractère C$ du mot ''MOSAIQUE'', il y a à la fois changements de PAPER et de PEN. Certains de ces petits carrés sont peu lisibles ! Qu'importe, l'essentiel était de vous montrer que toutes les fantaisies sont permises.

ON RESUME
a) On positionne le départ par un LOCATE (ou un TAB).
b) On change le PAPER ou le PEN ou les deux.
c) PRINT de caractères ou de blancs (SPC).
d) On revient à l'état de départ : PAPER 0 et PEN 1 (très prudent...).

Une petite remarque : En ligne 150, on a défini d'abord PAPER puis PEN. Si on fait l'inverse, il n'y a absolument aucun changement.

Par curiosité, tapez, en mode direct, MODE 0 et RUN. Les petits carrés du mot ''MOSAIQUE'' sont à la fois plus variés et plus lisibles. Remarquez aussi l'effet sur les numéros suivant ''PEN:''.

Revenez en MODE 1 et RUN. Entrez, en mode direct :

LOCATE 14,10:PRINT 12345 et Enter

Pourquoi ce désastre sur la bande bleu-ciel ? Parce qu'on a écrit 12345 (+1 blanc) avec le PAPER *en cours* (bleu foncé).

Tapez ensuite :

LOCATE 21,11:PEN 2:PAPER 3:PRINT ''I''

Là, on n'a pas fait de dégâts. On a remplacé le ''E'' de ''PAPER 3'' dans le bandeau rouge par un ''I'' bleu clair.

Remettez tout en ordre en tapant :

PAPER 0:PEN 1

Cette sage précaution...

LA GAMME COMPLETE

En MODE 0, il y a, par défaut, une sélection de 16 teintes sur les 27 disponibles, de ce fait il y en a onze méconnues. Ce court programme permet de vous les présenter en deux séries : n° de 0 à 15 puis n° 11 à 26. Les dernières sont vraiment pastel.

```
300 'LES 27 TEINTES
310 MODE 0:PEN 8
320 FOR RT=0 TO 15
330 INK RT,RT
340 PAPER RT:PRINT USING"##########";RT
350 NEXT
360 PAPER 1
370 PEN 8:INPUT"     TAPEZ ENTER",R$
400 ' SUITE
410 CLS:PEN 4
420 FOR RT=11 TO 26
430 INK RT-11,RT
440 PAPER RT-11:PRINT USING"##########";
RT
450 NEXT
460 END
```

Cela a amené l'auteur à rebaptiser certaines teintes. Voici notre nuancier :

0 noir
1 bleu foncé
2 bleu roi
3 rouge sombre
4 violet foncé
5 bleu violacé
6 rouge vif
7 pourpre
8 violet clair
9 vert
10 vert d'eau
11 bleu ciel
12 vert kaki
13 gris clair
14 bleu pastel
15 orange
16 chair
17 mauve pastel
18 vert fluo
19 vert pastel
20 turquoise clair
21 vert citron
22 vert tilleul
23 turquoise pastel
24 jaune vif
25 anis
26 blanc

Les teintes 19 et 22 sont très, très voisines, idem pour les numéros 20 et 23.

LA MEMOIRE COULEURS

Si vous avez modifié les valeurs par défaut des INK, de PAPER et PEN, elles resteront actives que vous fassiez RUN, NEW, CLEAR ou LOAD. Il est alors fcile de se faire piéger et l'on ne sait plus où l'on en est. C'est pourquoi, avant l'instruction END, un bon programmeur doit tout remettre dans l'état initial afin d'éviter des surprises au programme suivant... La ligne finale sera souvent du genre :

PAPER 0:PEN 1:MODE 1:END

Dans le programme précédent, les 16 INK ont été redéfinis (c'est exceptionnel), donc un seul remède, le radical SHIFT + CTRL + ESC.

Chapitre IX

LES BELLES PAGES D'ÉCRAN

C'est de la fioriture ? Non ! C'est un cachet.

Il y a deux sortes de programmes. Les petits passages expérimentaux pour voir ''si ça marche'' puis que l'on efface, et les LOGICIELS, c'est-à-dire un programme que l'on conserve pour s'en resservir plus tard, même s'il est court ; on risque aussi de le montrer à des amis. Là, le débraillé est inadmissible, ce programme mérite mieux que cela. Un logiciel sans présentation, c'est comme un bon vin servi dans un gobelet en plastique, ça gâche tout...

Qu'est-ce qu'un présentation minimum ?

Tout d'abord un titre, et bien centré ; puis des explications claires, exemple, le mode opératoire ou les règles du jeu car votre mémoire n'est pas infaillible. Donc, des phrases, elles aussi bien centrées, des lignes *espacées* et agréablement réparties sur la hauteur de l'écran, et non pas tassées en haut... En somme, la présentation minimum peut être sobre mais elle doit être PROPRE !

Deuxième degré, le plaisir des yeux, très important. Vous avez passé des heures, des jours à mettre ce programme au point, vous pouvez bien sacrifier un quart d'heure ou une demi-heure à lui conférer un abord sympathique. C'est ce même petit plus qui fait souvent la différence entre un ''bon restaurant'' et une ''gargotte''. Mais n'en faites pas trop quand même ! N'abusez pas des couleurs vives et clignotantes, des sons ou airs lancinants (comme dans certaines cassettes du commerce...). De la fantaisie, oui, mais pas de mauvais goût. Les possibilités couleurs, graphiques et sonores de l'AMSTRAD vous assurent un champ infini pour affirmer vos goûts artistiques.

PRESENTATION GENERALE

LE CENTRAGE

Servez-vous de la réglette de tabulation décrite au chapitre I, ou faites un centrage automatique en tabulant par (40 – LEN(A$))/2.

LES NUMEROS DE LIGNES D'ECRAN

N'utilisez jamais la ligne 1 si vous avez un BORDER différent de PAPER, car les lettres majuscules ainsi collées par le haut sont mal lisibles et inesthétiques.

Laissez une ligne vide entre deux lignes de texte (c'est super important).

Si vous remplissez l'écran, prenez garde au ''Ready'' qui va vous faire ''scroller'' l'écran de deux lignes vers le haut.

Séparez les ''rubriques'' par plusieurs lignes vides.

TYPES DE LETTRES

Tout en majuscules est pénible à lire. Réservez les majuscules pour les initiales ou pour un mot important dans une phrase. Il reste l'épineux problème des caractères accentués ''à, é, è ou ù'' ; il faudra se les faire, nous verrons cela en fin de chapitre.

CHOIX DES COULEURS

Il faut du contraste pour les lettres et bien sûr de la lisibilité'. Par contre, il faut peu de contraste entre deux PAPER voisins car c'est vite pénible à regarder. L'œil doit être attiré par le texte et non par le fond. C'est un fond différent qui met une phrase en évidence, il ne faut pas que ce soit l'inverse, sinon ça fait carnaval.

BORDER ou non ? Sans BORDER, l'écran paraît plus grand, avec, on fait ressortir la mise en page. Un BORDER est comparable au cadre d'un tableau afin de mettre ce dernier en valeur. On ne doit pas remarquer le cadre, donc évitez les couleurs criardes. Surtout pas la même couleur que PEN !

Les ophtalmies étant difficiles à soigner, n'utilisez qu'à bon escient les couleurs alternées clignotantes, par exemple afin d'attirer l'attention sur un *court* message très important ''perdu'' dans un écran déjà très chargé. Définir un INK juste pour cette inscription.

Lorsqu'un programme comporte plusieurs parties distinctes, il est conseillé d'établir pour chacune d'elles une présentation colorée différente ; le menu principal sera affiché avec un fond et des lettres spécifiques. Cela constitue un pense-bête pour l'utilisateur du proramme ; à la couleur dominante de l'écran, il sait dans quelle partie il se trouve. Ce sont des petits détails de ce genre qui créent le CONFORT d'utilisation d'un logiciel.

LES TRANSITIONS

Pour passer d'une page d'écran à une autre, on a le choix entre trois techniques :

— avec un INPUT, mais pas INPUT R$ qui laisse un ''?'' + le curseur, mais plutôt INPUT '''',R$ qui supprime le ''?'' ;
— par une boucle WHILE... WEND, déjà illustrée dans les programmes ''KEY'' et ''Réponse à un menu'' : pas de curseur d'attente, mais réagit avec une touche du clavier ;
— par une boucle d'attente programmée de quelques secondes. Donc pas de curseur ni d'intervention au clavier.

Nous attaquons à présent des techniques particulières.

— CALL &BB06

LES ANIMATIONS DE LETTRES

C'est facile et amusant, mais il ne faut pas en abuser, d'autant plus que l'AMSTRAD ne brille pas par sa vitesse d'affichage (à cause de la complexité de sa mémoire d'écran).

Le programe ci-dessous donne le mode opératoire pour six animations très différentes.

```
10 '----ANIMATIONS DE LETTRES----
20 '    EMPILEMENT PAR LA DROITE
30 CLS:BORDER 13
40 A$="Titre par ANIMATIONS de Lettres":
L=LEN(A$)
50 FOR I=1 TO LEN(A$):C$=MID$(A$,I,1)+"
"
60 FOR H=39 TO (40-L)/2-1+I STEP-1
70 LOCATE H,12:PRINT C$
80 NEXT:NEXT
100 '    EMPILEMENT PAR LA GAUCHE
110 FOR I=LEN(A$) TO 1 STEP-1:C$=" "+MID
$(A$,I,1)
120 FOR H=1 TO (40-L)/2-1+I
```

```
130 LOCATE H,14:PRINT C$
140 NEXT:NEXT
200 '    MONTEE DE TEXTE
210 FOR V=24 TO 16 STEP-1
220 LOCATE 1,V+1:PRINT CHR$(18)
230 LOCATE (40-L)/2,V:PRINT A$
240 NEXT
300 '    DESCENTE DE TEXTE
310 FOR V=2 TO 10
320 LOCATE 1,V-1:PRINT CHR$(18)
330 LOCATE (40-L)/2,V:PRINT A$
340 NEXT
400 '    APPARITION A GAUCHE
410 A$=SPACE$((40-L)/2-1)+A$
420 FOR I=1 TO LEN(A$)
430 LOCATE 1,6:PRINT RIGHT$(A$,I)
440 NEXT
500 '    APPARITION A DROITE
510 A$=A$+SPACE$((40-L)/2)
520 FOR I=1 TO LEN(A$)
530 LOCATE 41-I,20:PRINT LEFT$(A$,I)
540 NEXT
590 END
```

Vous constaterez que les translations verticales ne sont pas du meilleur effet sur l'AMSTRAD, donc à éviter.

NOTE : PRINT CHR$(18) efface la ligne. Ajoutez les lignes n° 65 et 125 contenant :

CALL &BD19

C'est plus joli, mais encore plus lent.

Faites EDIT 40, et remplacez A$ par quelque chose de plus court, ''AMSTRAD'', par exemple.

Comme il y a moins de lettres, certaines longueurs deviennent supportables, on gardera donc les lignes 65 et 125. D'autre part, vous remarquerez que le clignotement des translations verticales est plus acceptable pour une chaîne plus courte.

LES ENCADRES

Encadrer un titre par un trait est facile en utilisant les fonctions graphiques PLOT et DRAW ; nous l'avions fait dans le programme KEY, mais il y a d'autres variantes.

1) Placer le titre dans un rectangle d'une autre couleur, un ''bandeau''. Deux techniques : des bandes avec une autre valeur de PAPER comme au chapitre précédent (le ''PAPER 3'') ou placer le titre dans un WINDOW d'une autre couleur.
2) Encercler le texte avec des caractères graphiques quelconques.

Nous allons rassembler ces quatre méthodes dans un même programme, mais en paramétrant les tabulations afin que vous puissiez les adapter directement à vos propres programmes. Il est bien évident que, dans la pratique, vous n'aurez pas à recopier ces formules générales, vous mettrez directement les valeurs, ce qui raccourcira sensiblement la plupart de ces lignes. Ces formules vous éviteront néanmoins de longs tâtonnements.

```
10 '  ENCADRES DE TITRES
20 '  PAR UN TRAIT
30 MODE 1
40 A$=" TRAIT ":L=LEN(A$):H=(40-L)/2:V=4
50 LOCATE H,V:PRINT A$
60 PLOT H*16,(24-V)*16,2
70 DRAWR (L-1)*16,0
80 DRAWR 0,42:DRAWR -(L-1)*16,0:DRAWR 0,
-42
100 ' PAR BANDES
110 A$=" BANDE ":L=LEN(A$):H=(40-L)/2:V=
9
120 LOCATE H,V-1:PAPER 2:PRINT SPC(L);:P
APER 0
130 LOCATE H,V:PAPER 2:PEN 3:PRINT A$;:P
APER 0
140 LOCATE H,V+1:PAPER 2:PRINT SPC(L);:P
APER 0
150 PEN 1
200 ' PAR WINDOW
210 A$=" FENETRE ":L=LEN(A$):H=(40-L)/2:
V=14
220 WINDOW #1,H,H+L-1,V-1,V+1
230 PAPER #1,3:CLS #1:PEN #1,2
```

```
240 LOCATE #1,1,2:PRINT #1,A$
300 '  PAR CARACTERES
310 A$="  ETOILES  ":L=LEN(A$):H=(40-L)/
2:V=20
320 LOCATE H-1,V:PRINT "*";A$;"*"
330 LOCATE H-1,V-2:FOR I=1 TO L/2+2:PRIN
T "* ";:NEXT
340 LOCATE H-1,V-1:PRINT "*";SPC(L);"*"
350 LOCATE H-1,V+1:PRINT "*";SPC(L);"*"
360 LOCATE H-1,V+2:FOR I=1 TO L/2+2:PRIN
T "* ";:NEXT
370 END
```

Le facteur de 16, dans les lignes 60 à 80, peut vous intriguer : c'est pour la conversion de LOCATE en PLOT. En effet, 640/40 = 16, tout comme 400/25 ; on tient compte du fait que les progressions verticales en texte et en graphique sont en sens inverses en ligne 60 par (24 – V)*16.

Dans le programme ''WINDOW'', en ligne 230, nous incluons un CLS #1. C'est très important ! Quand on a défini un WINDOW, la première chose à faire est de l'EFFACER, souvenez-vous en...

Dans le programme ''caractères'', vous noterez qu'il faut séparer chaque caractère graphique par un blanc dans les lignes *horizontales.* Ceci pour obtenir un espacement voisin de celui des lignes verticales. Pour sortir de l'ordinaire, utilisez les caractères graphiques dont les codes ASCII se situent entre 123 et 255.

QUELQUES TRUCS

Pour souligner un mot, écrivez sous la ligne du dessous une suite de CHR$(208).

Pour afficher des guillemets, il y a deux méthodes, la classique PRINT CHR$(34) et la spéciale AMSTRAD : il suffit de remplacer la fonction PRINT par WRITE. Essayez :

```
WRITE ''NOM PROPRE''
```

La présence de virgules à la suite d'un PRINT provoque un espacement de 13 caractères entre chacune des données à afficher sur la même ligne ; ceci parce que, par défaut, on a ZONE 13. Pour avoir un autre espacement, par exemple 4, faire ZONE 4. A ce propos, vous avez le droit d'écrire :

```
PRINT A,USING''# # # '';B
```

Des PRINT CHR$ très utiles :
CHR$(17) efface la partie gauche de la ligne.
CHR$(18) efface la partie droite de la ligne.
CHR$(19) efface le haut de l'écran (ou WINDOW).
CHR$(20) efface le bas de l'écran (ou WINDOW).
CHR$(24) inversion PAPER/PEN (inversion vidéo).
CHR$(30) retour curseur en LOCATE 1,1.
CHR$(7) son BIP bref.
CHR$(1) ouvre le second jeu de caractères graphiques (de 1 à 31). Esayez :

```
FOR N=1 TO 31:PRINT CHR$(1);CHR$(N):'' ;:NEXT
```

LES CARACTERES AZERTY ACCENTUES

Nous allons redéfinir les sept caractères spéciaux du clavier AZERTY français, mais en respectant la norme ASCII internationale. De ce fait, ils seront transcrits de même sur une imprimante ''switchée'' ou commandée en clavier FRANCE. Il s'agit en fait d'un *sous-programme utilitaire* que vous allez enregistrer sur la cassette ''UTIL'' ; vous n'aurez ainsi qu'à faire un MERGE sur votre programme en cours d'écriture.

NOTE : Avant de taper ce programme, annulez les KEY DEF en cours en faisant SHIFT + CTRL + ESC.

```
51000 ' AZERTY ACCENTUE
51010 SYMBOL AFTER 7
51020 SYMBOL 64,96,48,120,12,124,204,118
,0
51030 SYMBOL 91,0,56,108,56,0,0,0,0
51040 SYMBOL 92,0,0,60,102,96,102,60,24
51050 SYMBOL 93,60,96,60,102,60,6,60,0
51060 SYMBOL 123,12,24,60,102,126,96,60,
0
51070 SYMBOL 124,48,24,102,102,102,102,6
2,0
51080 SYMBOL 125,48,24,60,102,126,96,60,
0
51100 RETURN
```

Recopiez cette légende sur un carton à ne pas perdre :
à = ∂ (code 64)
° = [(code 91)
ç = \ (code 92)
§ =] (code 93)
é = SHIFT + [(code 123)
ù = | (code 124)
è = SHIFT +] (code 125)

Ces définitions de touches, une fois lancées par un GOSUB 51000, sont résidentes, même après un NEW. Que se passe-t-il si on charge par dessus le programme KEY ? Il nous reste é, è et ù parce qu'il faut un SHIFT pour les obtenir, et = sur la touche TAB.

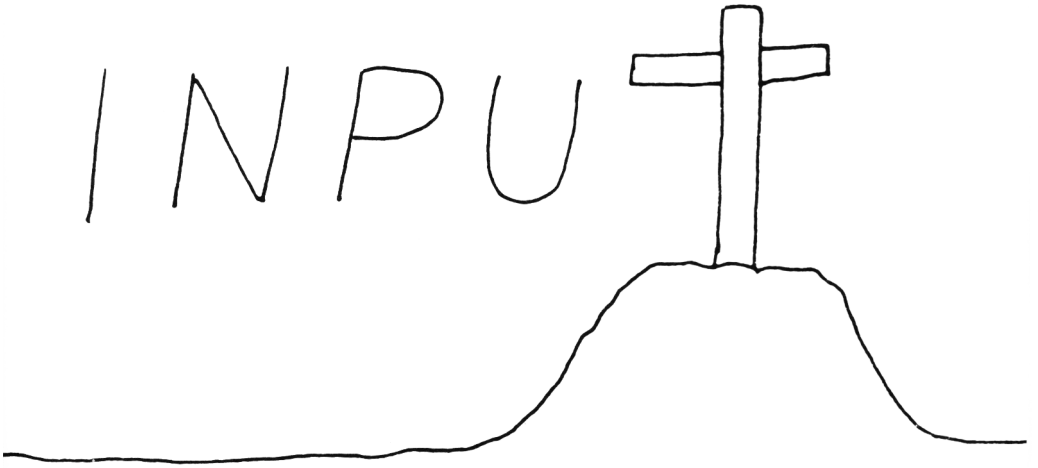
POUR CREER UN CARACTERE PAR SYMBOL

Si vous avez besoin d'un caractère très spécial ne figurant pas dans l'AMSTRAD, il est facile de le concevoir : sur papier cadrillé, tracez un carré de 8 par 8. Légendez les colonnes verticales de *droite à gauche* en partant de 1 et en doublant chaque fois, ce qui donne 1, 2, 3, 8, 16, 32, 64, 128. Facile ? Dessinez votre motif sur cette grille en petits carreaux ombrés.

Ensuite, on fait les comptes par lignes horizontales en partant du haut : 1re ligne, aucun carreau ombré, inscrire 0 en face de cette ligne. 2e ligne, il y a 3 carreaux ombrés, il s'agit des colonnes "32", "16" et "8" ; on fait le total 32 + 16 + 8 = 56 qu'on inscrit en face de la 2e ligne. Et ainsi de suite jusqu'en bas.

Il suffit alors de programmer SYMBOL suivi du code ASCII et de ces huit nombres. C'est ainsi que nous avons opéré pour définir nos sept caractères accentués.

Dernier problème : quel code ASCII utiliser ? On peut bien sûr ''taper'' dans les caractères graphiques que l'on appelle par PRINT CHR$(...), mais il reste encore quelques caractères ''libres'' accessibles par le clavier. Il s'agit de SHIFT + anti-slash, du signe Livre Sterling et de SHIFT + 0. Mais attention, ces caractères très spéciaux ne pourront pas apparaître sur imprimante !



Chapitre X

ENTREZ SANS PLANTER

Un programme bien construit est aussi une forteresse dont on ne peut trouver la faille, il doit être ''non-plantable''. Mettez-vous à la place d'un utilisateur qui, par étourderie, a répondu par une bêtise à un INPUT, cela arrive, mais si le programme se bloque en affichant un message d'erreur, l'utilisateur sera ''perdu'', et son travail antérieur aussi. Il aura le droit de dire ''le programmeur était un charlot !''.

Vous devez être en train de penser ''l'utilisateur, ce sera moi uniquement, et je connais bien mon programme''. Aujourd'hui, certes, mais pas quelques mois plus tard. Un oubli, une étourderie au clavier, et c'est le plantage honteux. Que faire alors ? RUN et tout recommencer ? Se plonger dans le listing afin de déterminer la manœuvre qui va remettre le programme sur rails ? Ni l'un, ni l'autre. Il fallait prévoir des SECURITES sur chaque entrée clavier.

Une ligne de sécurité s'écrit en moins de quinze secondes. Débloquer un programme prend en moyenne quinze minutes. L'étourderie fatale peut être reproductible, la sécurité elle est définitive. D'accord, les sécurités, ça prend des octets, comme les REM, mais on s'en moque, on a de la place sur l'AMSTRAD.

Essayez d'imaginer que l'utilisateur sera peu intelligent, étourdi et ignorant tout du Basic. Heureusement, tout cela est facile ; seulement quelques bonnes habitudes à prendre.

POUR ENTRER DES NOMBRES

Bannissez cette écriture de débutant :

PRINT ''QUANTITE?'':INPUT Q

A l'écran, vous avez le ''?'' et le curseur sous ''QUANTITE ?'', pas très esthétique. De plus, si on tape des lettres, on a au-dessus le message :

! Redo from start (=recommencez)

et une ligne encore au-dessous, le ''?'' et le curseur. Soit 4 lignes avec quatre ''?'', alors adieu à la belle page d'écran !

D'où la grande règle primordiale : pour entrer un nombre, on le demande sous forme de *chaîne,* et on le vérifie par VAL.

De l'entrée simpliste, passons sans transiter à l'entrée ''Super Luxe''. Il s'agit d'entrer une note sur 20.

```
10 '  ENTREE D'UN NOMBRE VERIFIE
20 MODE 1
30 LOCATE 13,8:LINE INPUT "Note sur 20:"
;NT$
40 IF INSTR(NT$,",") >0 THEN NT=-1 ELSE
NT=VAL(NT$)
45 IF NT$="" THEN 60
50 IF (NT=0 AND NT$<>"0") OR NT>20 OR NT
<0 THEN LOCATE 25,8:PRINT CHR$(10);CHR$(
7):GOTO 30
60 NT=ROUND(NT,1):PRINT NT
70 END
```

On décortique cela :

• Ligne 30 : La question est positionnée par un LOCATE, et grâce au '';'' l'entrée de la note est dans le prolongement du texte. En fait, on entre la note sous forme de chaîne NT$.

Pourquoi LINE INPUT au lieu de INPUT ? C'est dans le cas où, pour une note décimale, l'opérateur tape une virgule au lieu d'un point. Avec INPUT NT$, vous avez immédiatement droit à un ''Redo from Start'', pas avec LINE INPUT qui gobe tout.

• Ligne 40 : On recherche, par la fonction INSTR, la présence d'une virgule. Si oui, NT = – 1, valeur négative qui sera refusée à la ligne suivante ; sinon, NT = VAL(NT$).

• Ligne 50 : C'est le grand passage en douane. La première condition accepte la note 0, mais refuse l'entrée de lettres ayant provoqué par VAL un NT = 0. Remarquez les parenthèses qui lèvent toute ambiguïté. Seconde condition, il faut que NT soit compris entre 0 et 20, ce qui exclue le NT = – 1 de la ligne 40.

En cas de refus, l'entrée erronée est effacée par CHR$(18), on attire l'attention de l'étourdi par un BIP sonore et on revient à la ligne 30 pour reposer la question.

• Ligne 60 : On arrondit la note validée à une décimale, et on l'affiche. C'est bien sûr facultatif.

Il n'est pas facile de faire planter ce programme. L'auteur n'a trouvé que deux cas : les signes + ou − seuls ou suivis de caractères non numériques ; ce qui provoque un ''Type mismatch in 40''. La parade est possible mais lourde vis-à-vis de la faible éventualité.

On est obligé d'entrer une note, la touche Enter seule est refusée. Si cela ne vous convient pas, il suffit d'ajouter une ligne 45 :

45 IF NT$ = '''' THEN 70

LA SECURITE STANDARD

Vous devez penser ''Si je dois écrire tout cela à chaque INPUT !...''. Alors, créons un sous-programme de vérification vraiment standard (ligne 52000).

```
100 'SECURITE STANDARD
110 MODE 1
120 LOCATE 14,4:LINE INPUT "QUANTITE:",R
$
130 GOSUB 52000:IF FE THEN 120
140 QUA=R
150 LOCATE 9,7:LINE INPUT "PRIX unitaire
:",R$
160 GOSUB 52000:IF FE THEN 150
170 PU=R:SOM=QUA*PU
180 LOCATE 11,10:PRINT "PRIX TOTAL=";SOM
190 END
52000 ' VERIFICATION DES NOMBRES ENTRES
52010 FE=0:' FLAG D'ERREUR
52020 IF INSTR(R$,",")>0 THEN 52060
52030 IF (LEFT$(R$,1)="-" OR LEFT$(R$,1)
="+") AND VAL(MID$(R$,2,1))=0 THEN 52060
52040 R=VAL(R$):IF R$="" THEN 52070
52050 IF R=0 AND R$<>"0" THEN 52060 ELSE
 GOTO 52070
52060 FE=1:PRINT CHR$(7)
52070 RETURN
```

On remarquera que la variable entrée s'appelle toujours R$ (R comme réponse), c'est au retour du GOSUB que R est rebaptisé.

Nous faisons appel à ce que l'on appelle un FLAG (= drapeau) FE = flag d'erreur. Un flag est un témoin, un mouchard. On l'initialise à zéro en ligne 52010 : si une anomalie est constatée, il y a saut en 52060 où FE est mis à 1, plus signal sonore.

Dans le programme principal, il suffit de questionner le mouchard (lignes 130 et 160). Notez à ce propos que l'on simplifie l'écriture en IF FE au lieu de IF FE< >0. On aurait pu ajouter des clauses de rejet si R est négatif ou supérieur à telle valeur, vous savez faire !

Dans le sous-programme, on a profité de l'occasion pour inclure la ligne 52030 qui supprime le mini défaut du programme précédent (le + ou le −). Si cela ne vous convenait pas toujours, on peut faire un second module (copie) commençant en 52500, où à la fin de la ligne 52540, on aurait "... THEN 52560".

Après l'avoir testé, nous vous conseillons de faire DELETE − 190 et d'enregistrer le reste sur votre cassette UTIL.

LES CODES D'INTERVENTION

Supposons que l'écran demande d'entrer une suite de valeurs, mais l'opérateur est le seul à connaître leur nombre. Il faut donc prévenir l'ordinateur quand la "saisie de données" est terminée. C'est généralement la lettre Q (de quitter) qui donnera le signal stop. On doit aussi laisser une chance à l'étourdi qui s'aperçoit qu'il vient de faire Enter sous une valeur erronée. Il lui suffira de taper la lettre E (comme Erreur) et la valeur précédente sera "gommée". Mais il est important que ces codes d'intervention soient légendés dans un coin de l'écran.

A titre d'illustration, voici un programme tout simple qui calcule la moyenne d'âge d'un nombre *quelconque* de personnes.

```
10 ' MOYENNE D'AGES
20 ' NB=Nombre ; TA=Total Ages
30 CLS
40 LOCATE 9,21:PRINT "En cas d'ERREUR --
> E"
50 LOCATE 12,23:PRINT "Pour QUITTER --->
Q"
60 LOCATE 20,10:PRINT CHR$(10)
70 LOCATE 17,10:INPUT "AGE ",AG$
80 AG$=UPPER$(AG$)
90 IF AG$="Q" THEN 140
100 IF AG$="E" THEN NB=NB-1:TA=TA-AG:GOT
O 60
```

```
110 AG=VAL(AG$):IF AG=0 THEN PRINT CHR$(
7):GOTO 60
120 NB=NB+1:TA=TA+AG
130 GOTO 60
140 CLS:LOCATE 5,10:PRINT "L'age moyen d
e ces";NB;"personnes"
150 LOCATE 12,12:PRINT "est de";ROUND(TA
/NB,0);"ans."
160 LOCATE 1,21:PRINT CHR$(10):LOCATE 1,
23:PRINT CHR$(10)
170 END
```

On commence par afficher les légendes (lignes 40 et 50). En lignes 90 et 100, avant de valider AG$, on teste s'il ne s'agit pas d'un code. Si ce n'est pas le cas, on valide en ligne 110, puis on fait le bilan en ligne 120. On compte le nombre de personnes NB et on totalise les âges AG. On appelle ceci des ''incrémentations''.

Si le code est Q, on abandonne cette boucle pour afficher les résultats (lignes 140 et 150), et on efface les légendes (ligne 160) pour faire plus propre.

Revenons à la boucle, le code est ''E'' (ou ''e'', grâce à UPPER$). On défalque alors 1 de NB et la précédente valeur de AG (toujours en mémoire) au total d'âges TA. Remarquez ce que la petite ligne 100 apporte comme confort d'utilisation !

Deux autres petites remarques : On refuse les âges = 0 (évident...) et on arrondit la moyenne trouvée à l'entier le plus proche. Pensez bien à vous servir de la fonction ROUND avant d'afficher un résultat...

LES ENTREES DE CHAINES

Le seul ''problème'' avec les entrées de chaînes est leur formatage, c'est-à-dire imposer une longueur maximum ou fixe. C'est très simple, comme le montre le court programme ci-dessous qui calibre la chaîne entrée à 5 caractères, en ajoutant des blancs si nécessaire ou en prévenant par un BIP que la chaîne trop longue a été tronquée.

```
400 'CHAINE A LONGUEUR FIXEE
410 CLS
420 LOCATE 12,10:INPUT"NOM:",N$
430 IF LEN(N$)>5 THEN PRINT CHR$(7)
440 N$=LEFT$(N$+SPACE$(5),5)
450 LOCATE 16,13:PRINT N$,LEN(N$)
460 LOCATE 16,10:PRINT CHR$(10):GOTO 420
```

Une variante est de remplacer INPUT par INKEY$, ce qui permet de "surveiller" la chaîne à mesure qu'on la tape.

```
500 ' LETTRE PAR LETTRE
510 CLS:N$="":LOCATE 10,10:PRINT "PRENOM
:";
520 C$=INKEY$
530 IF C$=CHR$(13) THEN 570
540 IF C$<>"" THEN PRINT C$;: N$=N$+C$
550 IF LEN(N$)=6 THEN PRINT CHR$(7):GOTO
 570
560 GOTO 520
570 LOCATE 17,12: PRINT N$
580 END
```

La longueur maximum est ici fixée à 6. Au sixième caractère, c'est "entré" sans faire Enter. Pour un mot de moins de six lettres, on en sort par l'habituel Enter.

Cette technique est très pratique lorsque les zones d'entrée sont nombreuses sur une page d'écran, plusieurs sur la même ligne. C'est le cas du remplissage d'un tableau : si on écrit une chaîne trop longue avec INPUT, on risquerait d'abîmer la zone située à droite. Avec INKEY$, c'est impossible. Mieux, pour effacer une entrée erronée, on fait un PRINT SPC(6) au bon endroit, alors qu'un PRINT CHR$(18) effacerait tout ce qui est à droite.

LOGICIELS TABLEURS OU "CALC"

Ces logiciels établissent un tableau à deux dimensions (colonnes, lignes) aussi bien à l'écran qu'en DIM. Le tableau entré, on peut alors, en travaillant sur le DIM, faire des totaux, des moyennes, etc. sur telle colonne ou sur telle ligne.

Sans avoir recours à un tel logiciel du commerce, on peut très bien s'en faire un sur mesure, donc pour un problème bien particulier, un bilan annuel, par exemple.

Oublions, pour l'instant, la partie entrée en DIM, et parlons de l'écriture à l'écran de ce tableau (en MODE 2).

S'il y a 12 colonnes (12 mois), plus 17 lignes (17 clients ou fournisseurs ou postes de dépenses), il y aura donc $12 \times 17 = 204$ CASES à remplir (ou non). On reprend le petit programme ci-dessus, mais les coordonnées du LOCATE sont paramétrées, incrémentées dans des boucles FOR NEXT : le curseur passera donc automatiquement d'une case à l'autre et sans risque de mordre sur la case d'à côté. Ce serait un gros travail, mais déjà à votre portée, avec validations et codes d'intervention.

LE PLANTAGE INDIRECT

C'est celui qui peut se produire par combinaison d'entrées correctes. Un exemple : on a demandé d'entrer trois nombres N, A et B avec les sécurités habituelles, mais plus loin on définit C = N/(A – B) : Si A = B, on a un ''Division by zero error''...

Une fois votre programme terminé, scrutez le contenu de chaque ligne afin d'y débusquer tous les ''slashes'' (''/''). Puis, essayez d'imaginer tous les cas possibles où le diviseur peut être nul. Dans l'exemple ci-dessus, la sécurité serait simple, mais il existe des cas bien plus ''vicieux''. Au hit-parade des plantages surprises, c'est en effet la division par zéro qui arrive très largement en tête. Donc, soyez très vigilant. Un autre exemple concret : notre petit programme ''Moyenne d'âge''. Dès le départ du programme, tapez Enter, puis ''Q'' et Enter ; vous y aurez droit !

LA COMMANDE ON ERROR GOTO

A n'utiliser que dans les cas insolubles ou désespérés, car elle est super dangereuse à manier.

Sachez que, placée en début de programme, elle reste active pour n'importe quel type d'erreur, à n'importe quel endroit du programme.

Dès qu'elle a *fonctionné une fois,* elle est désamorcée, sauf si dans la ligne de destination il y a un RESUME vers l'endroit où s'est produit l'erreur (c'est un GOTO qui ré-active).

ERR renvoie le numéro de référence de l'erreur (voir pages A8.2 à 18.4 du manuel AMSTRAD).

ERL renvoie le numéro de ligne où s'est produit l'erreur.

RESUME ERL n'est pas admis (Syntax error).

Tout cela paraît complexe. En manipulant ce petit programme, vous saisirez les subtilités de ces quatre fonctions :

```
700 '   Emploi de ON ERROR GOTO
710 CLS
720 PRINT "Tapez + ou - pour erreur";PRI
NT
730 ON ERROR GOTO 8010
740 INPUT"MOT 1";R$
750 R=VAL(R$)
760 PRINT R$
770 INPUT"MOT 2";R$
780 R=VAL(R$)
790 PRINT R$
800 GOTO 740
810 END
```

```
8000 ' TRAITEMENT D'ERREUR
8010 IF ERL=750 AND ERR=13 THEN PRINT "P
AS VALABLE":RESUME 740
8020 PRINT "ERREUR NUMERO";ERR;"LIGNE";E
RL
8030 END
```

Après avoir activé la fonction en 730, on demande d'entrer deux fois un R$ (lignes 740 et 770). On provoquera une erreur (''Type mismatch'' = n°13) en tapant '' + '' ou '' − ''.

Après le END du programme principal (après, c'est très important), quelques lignes pour traiter l'erreur, celles adressées par le ON ERROR GOTO.

Là, on est très précis : cela ne concerne qu'une erreur de type 13 survenue à la ligne 750. Si c'est le cas, un message ''pas valable'' plus RESUME 740, c'est-à-dire le ON ERROR GOTO est réactivé puis ''GOTO'' 740 sur l'INPUT.

S'il s'agit d'une autre erreur, on passe donc à la ligne 8020 qui affiche la nature et l'origine de l'erreur, suivi d'un second END.

En lançant le programme, on constate que sur le ''MOT 1'', autant de fois que l'on tape '' − '', la ligne 8010 fait son office.

Tapons ''A'' ; au ''MOT 2'' tapons '' − '' : la fonction agit toujours et la ligne 8020 fait afficher ''Erreur n° 13 ligne 780'', et ''Ready'' à cause du END de la ligne 8030. La commande a agi, pas RESUME, elle est désamorcée. Et maintenant tapez :

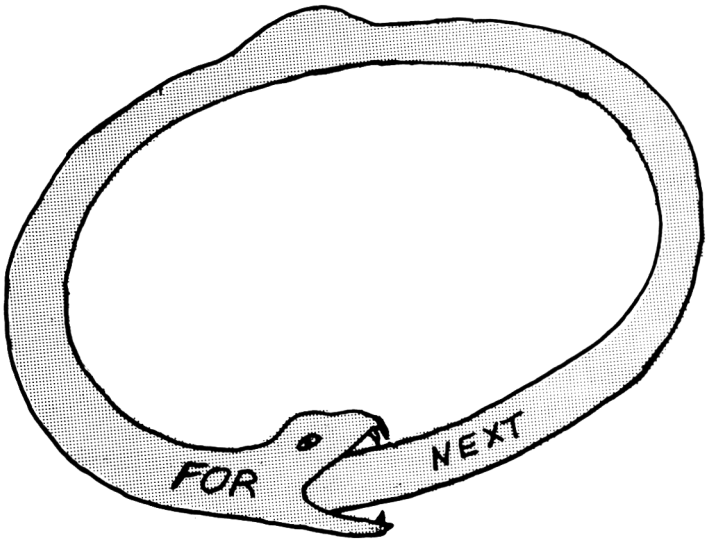
GOTO 740 et Enter

(on n'est pas repassé sur le ON ERROR GOTO). Au ''MOT 1'', tapez '' − '' : on a alors droit à ''Type mismatch in 750''.

Vous envisagez maintenant tous les ''retours de bâton'' possibles avec cette commande... Deux recommandations :

— mettre la ligne de correction d'erreur entre deux END, comme nous l'avons fait, et en précisant bien l'erreur (nature et ligne) ;
— ne jamais loger une boucle ON ERROR GOTO dans un *sous-programme ;* car si une erreur quelconque (et non envisagée) survient ailleurs, il y aura un saut dans ce sous-programme, sans GOSUB... Et là, ce sera le désastre !

CONCLUSION : A utiliser avec des pincettes.



Chapitre XI

LES BOUCLES ET LEURS PIÈGES

L'AMSTRAD dispose de trois sortes de boucles : les FOR... NEXT, WHILE... WEND et par GOTO numéro de ligne. Le choix se fait en fonction du problème à résoudre car chacune a ses avantages et ses inconvénients, un compromis entre vitesses, possibilités et sécurités d'emploi.

COMPARONS LES VITESSES D'EXECUTION

Le petit banc d'essai consiste à faire compter l'ordinateur de 1 à 10 000 en chronométrant.

```
10 ' VITESSES DES BOUCLES 1 A 10000
20 ' PAR FOR....NEXT
30 FOR I=1 TO 10000:NEXT:PRINT "FIN"
40 STOP:'temps=10.9 s
50 FOR N=1 TO 10000:NEXT N:PRINT "FIN"
60 STOP:'temps=11.1 s
70 FOR J%=1 TO 10000:NEXT:FRINT "FIN"
80 STOP:'temps=5.7 s !!!
90 '
```

```
100 ' PAR INCREMENTATIONS
110 WHILE K<10000:K=K+1:WEND:PRINT "FIN"
120 STOP:'temps=33.0 s
130 L=L+1:IF L<10000 THEN 130 ELSE PRINT
 "FIN"
140 STOP:'temps=33.0 s
150 ' ------------------------------------
160 ' RALENTISSEMENT DU A UN IF
170 FOR I=1 TO 10000:IF I=32000 THEN A=B
180 NEXT:PRINT "FIN"
190 STOP:temps=28.1 s au lieu de 10.9 s,
    soit 1.8 ms par IF.
```

De telles performances ont enchanté l'auteur. Non seulement l'AMSTRAD surpasse ses concurrents actuels en ridiculisant certains modèles japonais, mais on se retrouve avec des temps à peine supérieurs à ceux obtenus avec un IBM PC !

Commençons par les boucles FOR NEXT, ce sont les "formules 1". On constate que l'ajout (inutile) de la variable après NEXT ralentit légèrement. Si la variable est un nombre entier, la vitesse est presque doublée. Cinq secondes sept dizièmes, un record qui sera difficile à battre !

Par incrémentation, c'est-à-dire par + 1, on a le choix entre le WHILE WEND et le GOTO. Tous les deux ont la même vitesse, mais attention. Trois fois plus lents que par FOR NEXT... En mettant une variable entière (K% et L%), le temps descend à 25,2 secondes, c'est mieux mais pas très spectaculaire par rapport à 33 s.

Dans des boucles, il y a très souvent des conditions IF. En ligne 160, nous avons voulu montrer leur effet ralentisseur. La comparaison très simple de la ligne 170 prend 1,8 ms ; pour des comparaisons plus complexes, on peut atteindre dix fois plus ! Donc, soyez assez radin sur les IF dans une boucle répétée de nombreuses fois.

Vous pensez certainement que des boucles de 1 à 10000 ne se rencontrent jamais et que, de ce fait, vous n'aurez jamais un "pédalage" de trente secondes. Sur une seule boucle, c'est vrai, mais trois ou quatre *boucles imbriquées* est chose courante, et si vous faites le calcul, vous constaterez que la boucle "centrale" fait des milliers de passages et avec calculs et des IF... Exemples : certains programmes pourtant courts produisant des figures graphiques compliquées et qui demandent plusieurs minutes...

Le temps d'exécution total constitue donc le premier piège : en reprenant le programme, en remplaçant des boucles GOTO par des FOR NEXT, en mettant des variables entières, en ajoutant un IF qui peut en court-circuiter cinq autres, on arrive souvent à ramener un pédalage de dix secondes (très pénible) à deux secondes seulement. Le plus comique est que le listing s'en trouve généralement raccourci lui aussi, mais il a fallu le repenser entièrement, et souvent à plusieurs reprises. A considérer comme un jeu intellectuel, comme les échecs, mais là, il en reste quelques chose.

IMPORTANT : Quand vous savez qu'une opération en boucles peut durer plusieurs secondes, il faut en prévenir l'utilisateur par un petit message à l'écran du genre ''Patience... ''. Sinon, l'opérateur croit à un plantage et peut faire un Break ! Si ce passage long est fréquent, rompez la monotonie en préparant quelques messages d'attente en DIM MESSAGE$; c'est une fonction RND qui fera le choix. Un peu d'humour sera le bienvenu.

LES PIEGES DES BOUCLES FOR NEXT

La coupable est presque toujours la valeur du STEP. Nous avions vu au chapitre VII la farce que peut nous jouer un STEP décimal, le célèbre :

```
FOR I=0 TO 1 STEP 0.05
```

où la valeur n'est jamais atteinte (farce annulée par la fonction ROUND). Les grands dangers sont avec les STEP obtenus par un calcul et pouvant conduire à des valeurs imprévues, donc à prévoir. Exemple :

```
FOR I=1 TO 328000 STEP-1
```

C'est instantané ! Aucun passage, on sort néanmoins de la boucle et I=1. Tout se passe comme si la boucle FOR NEXT et son contenu n'existaient pas. C'est assez traitre...

CONCLUSION : Quand un STEP est paramétré, il faut prendre les mêmes précautions qu'avec un nombre diviseur.

Rappelons qu'après le NEXT, la variable est égale à sa dernière valeur + un STEP. Ainsi, après un FOR I=0 TO 10:NEXT on a I=11 et non pas 10. Un autre exemple rassurant :

```
FOR I=1.3 TO 5:PRINT I;:NEXT
```

On obtient 1.3 2.3 3.3 et 4.3. Si après on fait PRINT I, on a I=5.3. Vous constatez donc qu'il n'est pas obligatoire que la valeur cible soit atteinte.
Pour les sorties de boucles prématurées, les règles sont les mêmes que pour les sous-programmes, à savoir qu'il n'y a qu'une seule et unique ligne où figure l'unique NEXT. Un GOTO vers cette ligne sera le raccourci.

Si on peut sortir d'un sous-programme par un GOSUB, on ne peut sortir d'une boucle FOR NEXT par un GOTO, sinon gare à la casse...

Deuxième cas : vous introduisez une condition qui doit précipiter la fin de la boucle, voici un exemple schématique.

```
600 FOR I=1 TO 360
- - - - - - - - - - -
650 IF A=N THEN I=360:GOTO 680
- - - - - - - - - - -
680 NEXT
```

C'est garanti sans danger. Si vous désirez connaître la valeur de I à cet instant, mettez un flag FI.

```
650 IF A=N THEN FI=I:I=360:GOTO 680
```

mais pensez à initiliser FI à 0 avant le FOR.

LES PIEGES DES BOUCLES A INCREMENTATIONS

C'est-à-dire avec un ''compteur'' genre K = K + 1 inclus dans une boucle GOTO ou dans un WHILE WEND.

Le piège le plus classique est la variable non initialisée à zéro. En effet, si précédemment vous avez utilisé ce même nom de variable dans une boucle FOR K = 1 TO 12, vous risquez d'être désorienté parce que le comptage démarre à 13 + 1 = 14 au lieu de 1...

Le deuxième piège est une condition de fin de boucle trop précise, du genre IF K = 10 car s'il ''arrive'' K = 9.999999, il n'y aura pas de fin de boucle... Deux vaccins contre cela : la fonction ROUND ou utiliser les opérateurs < ou > ou <= plutôt que le signe =.

Pour des sorties prématurées, même technique, renvoi au numéro de ligne où figure le GOTO de fin de boucle. Cela fera deux GOTO à la file, mais c'est sans ennuis.

L'avantage du WHILE WEND sur le GOTO est sa clarté. On voit la boucle dans un listing, on voit la ligne *commençant* par WHILE, et plusieurs lignes plus bas on cherche et on trouve la ligne WEND. C'est net et sans ambiguïté ; ce qui est loin d'être le cas avec les GOTO ! Pour la clarté, prenez un numéro de ligne multiple de 100 pour un départ de boucles.

Il n'est pas possible d'imbriquer des boucles WHILE WEND. Avec des GOTO, c'est techniquement possible mais vivement déconseillé car vous aurez du mal à vous y retrouver ; autant essayer de démêler 50 mètres de fil de pêche...

Faites attention à l'endroit où vous placez votre compteur, en début ou en fin de boucle. Exemple : dans le programme ''Moyenne d'âge'' du chapitre précédent, les compteurs sont en ligne 120 APRES les validations d'entrée.

UNE COMPTABILITE D'ERREURS

Le petit exemple qui suit illustre nos dernières recommandations. Il s'agit d'un sous-programme gadget qui va comptabiliser les étourderies de l'opérateur. Il sera appelé depuis notre programme "Sécurité standard" du chapitre X ; il suffira de modifier ainsi sa dernière ligne :

```
52070 GOSUB 8000
52080 RETURN
```

```
8000 ' COMPTAGE DES ERREURS COMMISES
8010 IF FE=0 THEN 8080
8020 TE=TE+FE :' TOTAL ERREURS
8030 IF TE<5 THEN 8080
8040 IF TE>=5 AND TE<10 THEN LOCATE 4,24
:PRINT "DEJA";TE;"ERREURS,faites attenti
on...";CHR$(7)
8050 IF TE>9 THEN LOCATE 1,24:PRINT TE;"
ERREURS! Prenez un peu de repos...";CHR$
(7)
8060 FOR I=1 TO 3000:NEXT
8070 LOCATE 1,24:PRINT CHR$(10)
8080 RETURN
```

A partir de la cinquième erreur et jusqu'à la neuvième, apparaît un message en bas de l'écran avec bip sonore. Au-delà, c'est un autre message...

On remarque en 8010 que, si la réponse a été correcte (FE = 0), on quitte directement, sinon on compte, en 8020. Avec TE de 1 à 4, on ne dit rien et on sort.

La ligne 8060 est une boucle d'attente d'environ trois secondes, puis on efface le message en bas d'écran (ligne 8070).

QUELLE BOUCLE CHOISIR ?

La FOR NEXT a tout pour plaire. C'est de très loin la plus rapide, c'est clair dans le listing, on peut imbriquer à volonté sans le moindre risque. Un seul défaut : il faut définir, dès le départ, le nombre de cycles à effectuer, bien que nous ayons vu une échappatoire, mais il fallait quand même fixer un maximum.

Les incrémentations sont beaucoup plus lentes, mais il n'est pas obligatoire d'avoir au départ une idée du nombre de cycles (voir ''Moyenne d'âge''). Autre avantage : le pas (équivalent du STEP) peut varier d'un cycle à l'autre, bien que le cas soit assez rare. Inconvénient majeur : on ne peut les imbriquer.

Enfin, pour le WHILE WEND, n'oublions pas son association à INKEYS pour l'attente d'une touche.

```
53000 ' ATTENTE D'UNE TOUCHE
53010 LOCATE 0,24:PRINT "Pressez une Tou
che,SVP ..."
53020 WHILE INKEY$="":WEND
53030 LOCATE 0,24:PRINT CHR$(18)
53040 RETURN
```

A enregistrer sur cette préciseuse cassette UTIL.

Chapitre XII

LA GESTION DES DONNÉES DIM, DATA, FICHIER

Tout programme comprend des ''données'', depuis de simples constantes genre N = 124 ou A$ = ''JEAN'' jusqu'à de longues séries de valeurs ; soit fixées dans le programmes, soit entrées par l'opérateur répondant à des INPUT, ce sont alors des données ''saisies''.

D'autre part, il faut savoir que le magnéto-cassette ne sert pas uniquement à faire SAVE ou LOAD pour des programmes, il sait aussi enregistrer et lire des ''fichiers''. Ces enregistrements sont constitués de suites de nombres et de chaînes, des centaines, des milliers !

A partir de là, on peut dire qu'il y a quatre sortes de programmes :

1) Ceux où les données sont immuables, en nature et en nombre. Elles sont écrites une fois pour toutes dans le programme afin d'y être utilisées.
2) Les données sont entrées une à une par l'opérateur, puis le programme les traite puis affiche le ou les résultats à l'écran, et c'est terminé. C'est le cas des programmes de calcul complexes.
3) Là aussi, les données sont saisies mais ne sont pas traitées. Elles sont ensuite enregistrées sur cassette (ou disquette) afin de compléter un fichier enregistré. C'est un ''programme de saisie''.
4) C'est le progamme qui lit et qui exploite ce fichier enregistré. C'est un ''programme de traitement de fichier''.

NOTE : Il exite dans le commerce des programmes dits ''gestionnaires de fichiers'' qui regroupent les fonctions saisie et lecture-exploitation. Il en existe sur cassette, mais pour une utilisation professionnelle, il faut deux lecteurs de disquettes et un logiciel valant souvent plusieurs milliers de francs.

Lorsque les données sont des nombres que l'on ''range'' dans un tableau afin de faire des calculs (bilans, etc.), ces logiciels s'appellent des ''tableurs'' ; souvent leurs noms commerciaux se terminent par ''CALC''.

Fermons cette parenthèse sur ces logiciels professionnels, ou ''Proriciels'', conçus pour des non-informaticiens, et revenons à notre programmation en Basic.

LES TABLEAUX DIM

La fonction DIM est un cas à part. Etant débutant, on en apprend le principe de base, on l'utilise constamment, mais sans chercher à en savoir plus sur elle. Conséquence : on ne soupçonne pas ses autres possibilités ni ses pièges...

L'INITIALISATION

C'est au début du programme qu'il faut initialiser *tous* les tableaux DIM, et ce *sur la même ligne.* Exemple :

```
50 DOM BR(15,3),AC$(18),ME%(340),J(6)
```

Cette pratique a trois avantages :

— On n'a pas à ré-écrire DIM à chaque fois.
— C'est plus clair et plus pratique dans un listing.
— Le programme ne risque pas, par un GOTO, de repasser sur cette ligne ; car définir une deuxième fois un DIM (même sans rien changer) provoque un plantage avec message d'erreur.

Dès le départ du programme, le BASIC réserve de la place pour ces tableaux, même s'ils sont encore vides. Comptez deux octets par ''case'' de nombres entiers, trois octets pour des chaînes, et cinq octets pour des réels.

La preuve, après avor fait NEW, tapez :

```
10 DIM A(840,10) et RUN
```

vous aurez le message ''Memory full in 10''.

En effet, $841 \times 11 \times 5 = 46255$, alors que l'AMSTRAD dispose de 43533 octets.

Nous avons bien calculé 841 × 11 et non 841 × 10 car l'indice zéro existe ! Ainsi, dans un DIM B(12), vous avez 13 ''places'', dans DIM C(12,1) vous en avez 26, et dans DIM D(12,2) il y en a 39 et non 24 ! Ne gaspillez pas ces octets, leurs places sont réservées en RAM, alors utilisez-les.

Vous trouvez surprenant que l'on définisse DIM J(6) alors que ce n'est pas obligatoire jusqu'à 10 : économie d'octets ; sinon, dès que le Basic rencontre une ''variable indicée'', genre J(1) = 24, sans vous prévénir, il crée lui-même DIM J(10), d'où un (petit) gaspillage de 4 × 5 = 20 octets.

LA COMMANDE ERASE

Elle a deux effets simultanés. Elle efface toutes les valeurs du DIM concerné (exemple ERASE B), elle efface l'initialisation du DIM. On peut alors le redéfinir avec d'autres DIMensions. Donc, après avoir vidé un tableau pour le réutiliser, pensez à réinitialiser ce DIM après ERASE. Parfois, on ERASE un DIM encombrant dont on n'a plus besoin, uniquement pour faire le ménage dans la RAM.

LES DIM MULTI-DIMENSIONS

On se représente facilement un DIM BILAN(12,4), cinq ''colonnes'' sur treize ''lignes'' ; c'est, par exemple, quatre postes de dépense sur les douze mois, les colonnes et lignes d'indices zéro étant réservées pour les totaux de lignes et de colonnes. Mais que représente DIM BILAN (12,4,3) d'une façon concrète ? Le même tableau que le précédent, mais à plusieurs ''étages'', soit quatre feuilles l'une sur l'autre représentant chacune un tableau 13 × 5. Numéro 1 = bilan 83, numéro 2 = bilan 84, numéro 3 = bilan 85 et numéro zéro = moyennes ou totaux. Et DIM BILAN %(12,4,3,2) ? Trois piles de quatre tableaux de cinq lignes sur treize colonnes. Et cela ne fait que 13 × 5 × 4 × 3 × 2 = 1560 octets ou 3900 octets (environ...) avec des nombres réels.

Calculez toujours vos encombrements ; des fois il y a des surprises...

LA FONCTION DATA

DATA, en début de ligne, annonce une suite de noms, de nombres, même mélangés. Cette ligne *ne peut être lue que par la fonction READ* (très important pour la compréhension).

Quand le déroulement du programme arrive sur une ligne de DATA, il *l'ignore* comme s'il s'agissait d'une ligne de REM.

De ce fait les lignes DATA peuvent être éparpillées dans le listing (ce n'est pas recommandé pour la clarté) ; même après le END.

Elles seront lues par READ dans l'ordre de leurs numéros de lignes. On peut ainsi imaginer qu'elles sont rassemblées dans un coin de la RAM.

Voyons d'abord la syntaxe d'une ligne de DATA. Tout est permis, pourvu que les données soient séparées par une virgule. Ainsi, des nombres peuvent être interprétés comme nombre réels, entiers ou même chaînes, tout dépend de la variable suivant READ. Exemples : READ N ; READ J% ou READ A$.

On ne mettra des guillemets que pour inclure des blancs ou des virgules. Exemple :

100 DATA JEAN,'' PAUL '','',PIERRE'','',''

(cette ligne contient quatre DATA).

Le nombre de données par ligne est sans importance, on vous recommande par contre d'en mettre le même nombre à chaque ligne ; cela simplifie beaucoup les vérifications.

Parlons à présent de la commande RESTORE, et pour cela reparlons de la ''pile'' de lignes de DATA. Supposons que ces lignes ont pour numéros 110, 120, 130, 460, 520 et 610. On parle alors du ''pointeur'' de DATA, dès le RUN il est, par défaut, sur ''110'', la première ligne. A mesure que READ lit les données, le pointeur se déplace le long de chaque ligne et de ligne en ligne. Quand le pointeur a désigné la dernière donnée de la dernière ligne, il n'y a plus rien à lire par READ, sinon on aurait plantage avec le message ''DATA exhausted in 420'' (420 étant la ligne où se trouve le READ).

Pour recommencer la lecture, il faut ''remonter'' le pointeur, c'est le rôle de RESTORE. Tout seul, le pointeur est mis sur la première ligne, ce serait ici la même chose que RESTORE 110. On peut imposer le pointeur sur une ligne quelconque, par RESTORE 460, puis le pointeur enchaînera sur 520 et 610.

Le petit programme qui suit illustre la souplesse des instructions DATA, READ et RESTORE.

Nous avons placé une ligne de DATA après le END afin de vous montrer que son emplacement n'a aucune importance.

```
10 '  CELEBRITES
20 CLS:PRINT
30 DATA VICTOR,HUGO,1802,1885,JEAN,RACIN
E,1639,1699
40 FOR I=1 TO 4
50 READ PR$,NOM$,NAIS,MORT
60 PRINT PR$;" ";NOM$;" est mort a";MORT
-NAIS;"ans."
70 NEXT
80 PRINT
100 RESTORE 150
110 FOR I=1 TO 5:READ X$:NEXT
120 READ A$,B$,C$
```

```
130 PRINT A$;" (";B$;"-";C$;')"
140 END
150 DATA HECTOR,BERLIOZ,1803,1869,LOUIS,
PASTEUR,1822,1895
```

CELA DONNE:

```
VICTOR HUGO est mort a 83 ans.
JEAN RACINE est mort a 60 ans.
HECTOR BERLIOZ est mort a 66 ans.
LOUIS PASTEUR est mort a 73 ans.
PASTEUR (1822-1895)
```

En ligne 50, on "READ" quatre données à la file et les désigne comme étant chaînes ou nombres.

En ligne 100, on "remonte" le pointeur à 150, on lit "à blanc" les cinq premières données de la ligne qui ne nous intéressent pas, afin de prendre les trois suivantes comme variables chaînes. On constate donc que les données 1822 et 1895 peuvent fournir indifféremment des nombres ou des chaînes.

En résumé, pour "sortir" une donnée en DATA, faire RESTORE sur sa ligne, et lire à blanc les données qui précèdent.

DIM ET DATA COMBINES

On aurait encore davantage de souplesse d'exploitation si les données précédentes étaient disposées dans une tableau DIM C$(4,4), quatre noms, quatre rubriques. Mais il serait très fastidieux d'écrire : C$(1,1) = "VICTOR":C$(1,2) = "HUGO": etc, alors qu'il est si simple de remplir le DIM par des FOR NEXT lisant les data :

```
FOR I=1 TO 4:FOR J=1 TO 4:
READ C$(I,J):NEXT:NEXT
```

C'est tout...

LES FICHIERS SUR CASSETTES

Pour ne pas vous perdre dans les définitions de OPENOUT, OPENIN, CLOSEOUT, CLOSEIN, rappelez vous que OUT et IN sont, par rapport à la RAM, au micro-ordinateur, le magnétophone est lui *extérieur,* même si on l'a loge physiquement sous le même capot. OPEN = ouvrir, débuter ; CLOSE = fermer, terminer. Donc OPENOUT signale le début d'une sortie (vers le magnéto-cassette), c'est donc un début d'enregistrement.

Inversement, et par le même raisonnement, CLOSEIN est la fin d'une lecture de fichier sur cassette. OK ? Illustrons tout cela par ce petit programme :

```
300 ' FICHIER SUR CASSETTE
310 '  ENREGISTREMENT
320 CLS:SPEED WRITE 1
330 DIM N(12)
340 FOR I=1 TO 12
350 N(I)=I*30
360 NEXT
370 LOCATE 2,10:PRINT "PREPAREZ UNE CASS
ETTE POUR ENREGISTRER"
380 LOCATE 12,12:PRINT "LE FICHIER ";:WR
ITE "SERIE"
390 GOSUB 600
400 OPENOUT "SERIE"
410 FOR I=1 TO 12:WRITE #9,N(I):NEXT
420 WRITE #9,"CECI EST UNE SERIE DE VALE
URS:"
430 CLOSEOUT
440 CLS
450 LOCATE 15,10:PRINT "REWIND SVP..."
460 GOSUB 600:CLS
470 '-----REWIND-----
500 ' LECTURE - CHARGEMENT DU FICHIER
510 OPENIN "SERIE"
520 INPUT #9,A,B,C,D,E,F,G,H,I,J,K,L,A$
530 CLOSEIN
540 CLS:LOCATE 10,5:PRINT "CONTENU DU FI
CHIER LU:":PRINT :PRINT
550 PRINT A$;A;B;C;D;E;F;G;H;I;J;K;L
560 END
600 ' ATTENTE
610 LOCATE 7,16:PRINT "...puis pressez u
ne Touche"
620 WHILE INKEY$="":WEND:PRINT
630 RETURN
```

Dans un premier temps, nous remplissons un tableau DIM N(12). Préparez une cassette vierge (avec amorce avancée). Remarquez qu'en ligne 380 nous utilisons WRITE au lieu de PRINT pour afficher ''SERIE'' entre guillemets à l'écran, histoire de vous rappeler que WRITE est un PRINT ''qui prend tout''.

On commence l'enregistrement du contenu du tableau en 400. OPENOUT suivi du titre vous rappelle la syntaxe du SAVE, d'ailleurs, le même message ''Press REC and PLAY then any key:'' apparaît à l'écran. Mais ensuite il faut ''éjecter'' les données vers le receveur, c'est le rôle de WRITE #9 (ligne 410), même syntaxe que PRINT.

En ligne 420, nous décidons d'ajouter une longue chaîne à l'enregistrement. Terminé, donc CLOSEOUT, le magnéto-cassette s'arrête, le fichier ''SERIE'' est enregistré.

Manœuvre inverse. La récupération de ce fichier (ligne 500). D'abord REWIND, cela va de soi.

Ligne 510 OPENIN ''SERIE'' rappelle la syntaxe de LOAD. Même message à l'écran. Mais il faut ''injecter'' ces données par INPUT #9 (ligne 520) ; vous remarquerez que l'on peut rebaptiser les données à condition de rester fidèle à la fois à leurs natures (nombre ou chaînes) et à leur quantité. Et on ferme par CLOSEIN.

PETIT RAPPEL : La commande INPUT ordinaire, par le clavier, est en fait INPUT #0, le canal 0 étant le clavier, #9 le magnétophone (=INPUT par #0 ou #9). Même remarque pour PRINT (=PRINT #0) et WRITE (=WRITE #0).

Ceci fait, nous prouvons, en ligne 550, que nos valeurs ont bien été rechargées.

LA FONCTION EOF (=END OF FILE = fin de fichier)

Elle ne sert qu'à la lecture. Il est des cas où le programme chargé de récupérer un ficher ignore le nombre de données qu'il renferme. En ce cas, il est obligatoire que toutes les données soient de même nature (entiers, réels ou chaînes).
Exemple :

```
710 OPENIN ''MACHIN''
720 INPUT #9,N$
730 PRINT N$;:K=K+1
740 IF EOF THEN 800
750 GOTO 720
800 CLOSEIN
810 PRINT ''Il y avait'';K;''données''
```

On aurait pu aussi charger un tableau DIM préalablement surdimensionné par précaution, par exemple, DIM N$(250). Les lignes 720 et 730 devraient alors être modifiées comme suit :

```
720 K=K+1
730 INPUT #9, N$(K)
```

Si, dans les lignes 410 et 420 vous remplacez WRITE #9 par PRINT #9, vous constaterez, en relançant tout le programme, que ça marche aussi bien. Mais WRITE est plus sécurisant.

Si vous avez une imprimante, vous pourrez remplacer WRITE #9 par WRITE #8 ou PRINT #8. Les données sortent alors vers l'imprimante et non plus vers la cassette.

NOTE : En ligne 320, nous nous mettons en ''grande'' vitesse d'enregistrement. Quoiqu'en dise le manuel, il faudrait des cassettes d'une qualité vraiment épouvantable pour être contraint d'enregistrer en petite vitesse.

Chapitre XIII

LA CHASSE AUX BUGS

Un ''bug'' ou ''bogue'' est une erreur dans un programme. Il en existe deux espèces, le bug franc qui fait planter le programme avec ou sans message d'erreur, et le bug fantôme, très sournois, qui laisse croire que tout se déroule bien et qui, une fois de temps en temps, provoque un plantage, ou ne plante pas mais fournit des résultats erronés (mais pas toujours flagrants...) ; de ce fait, on ne remarque la présence de cette ''sale bête'' que bien plus tard. Heureusement, cette seconde espèce est bien plus rare.

Avant de passer aux méthodes de dépistage systématiques, nous allons passer en revue les erreurs les plus classiques.

LES ETOURDERIES DE SYNTAXE

On n'a pas à la chercher puisque l'écran EDITe cette ligne : espace, lettre, paramètre oubliés sont de loin les cas les plus fréquents ; tout de suite rectifiés. Quand vous utilisez une fonction Basic dont vous n'avez pas encore l'habitude, vous risquez un ''Syntax Error'' ou un ''Improper Argument''. Un coup d'œil sur le manuel vous renseignera ; d'où une précaution élémentaire : ne mettez pas sur une même ligne plusieurs clauses (séparées par '':'') contenant des mots Basic dont vous n'êtes pas sûr.

Et puis il y a l'erreur tellement bête que l'on ne la remarque pas du premier coup, c'est la lettre ''O'' à la place d'un zéro, virgule à la place d'un point, un anti-slash à la place d'un slash, un ''l'' minuscule confondu avec un ''1''. Ne souriez pas, cela arrive encore même après des années de pratique...

DES ERREURS DE SYNTAXE PLUS SUBTILES

Elles sont beaucoup plus rares sur l'AMSTRAD car son Basic exige que le libellé d'une fonction ou d'une commande soit séparé par une espace (ou '' ou #). De ce fait, on peut mettre des noms de variables contenant des mots réservés. Ainsi, IFG = 45 et correct, alors que bien d'autres micros liraient IF G = 45.

Si vous connaissez mal la liste des ''mots réservés'' de l'AMSTRAD, vous risquez d'utiliser l'un d'eux comme nom de variable, ainsi LET = 26 ou LET$ = ''ABC'' feront erreur parce que LET est un mot-clé, ou réservé. Bien que totalement inutile, il existe.

La liste *complète* de ces mots est celle des pages A8.5 et A8.6 du manuel d'origine. Il n'en manque qu'un seul : DEC$. D'après le ''AMSTRAD CONCISE BASIC SPECIFICATION SOFT 157'', page D.2, cette fonction transformerait un nombre en chaîne formatée, comme PRINT USING. Là, il y a un bug dans la ROM, car ça ne marche pas ; c'est sans doute pour cela que le manuel d'origine en français n'en parle pas. Il n'empêche que le fait d'écrire DEC$ = ''PAUL'' provoque un ''Syntax Error'' !

Quand le mot Basic se termine par $, vous pouvez utiliser la partie lettres comme nom de variable. Ainsi, CHR = 123 ou STR = 245 sont admis, mais l'inverse du genre LET$, DIM$ ou FOR$ est interdit. Si vous y tenez, il suffit d'ajouter une lettre : LETT, DIME et FORT sont des noms admis.

LES FARCES DE LA LOGIQUE

Les fonctions logiques sont les suivantes :

IF, THEN, ELSE, AND, OR, XOR et NOT

Elles sont très fréquentes (sauf XOR et NOT) et d'un fonctionnement très *strict,* donc elles conduisent ainsi à des réactions logiques autres que celles envisagées, et sans message d'erreur.

Voici l'exemple de la confusion classique : en réponse à un menu, vous voulez qu'il y ait un bip sonore pour les réponses ''B'' *et* ''E''. Alors, traduisant le français en Basic, vous tapez :

```
IF R$ = ''B'' AND R$ = ''E'' THEN PRINT CHR$(7)
```

Condition impossible à remplir ; il fallait écrire OR et non AND.

L'autre erreur classique est de mettre plusieurs clauses sur une ligne en oubliant que si un IF n'est pas satisfait, la suite de la ligne est ignorée :

```
70 INPUT R$
80 IF R$ = "JEAN" THEN A = 5:CLS
```

Si R$ n'est pas JEAN, le CLS ne sera pas exécuté.

On a parfois affaire à des conditions logiques fort complexes, en ce cas, n'hésitez pas à mettre des parenthèses :

```
IF (CARACT$ = "bête" AND RICH$ = "pauvre") OR (FIG$ = "laid" AND
    ASPECT$ = "négligé") THEN REPONSE$ = "Du Vent!!"
```

Avec ces parenthèses, c'est à la fois clair pour l'homme et pour le Basic ; pas d'ambiguïté possible.

Essayez ce petit programme qui illustre les fonctions logiques et leur comportement.

```
10 ' VALEURS LOGIQUES
20 PRINT " ENTREZ 2 NOMBRES DE 1 A 5"
30 INPUT "A";A
40 INPUT "B";B
50 VERIF = (A=2 AND B=3)
60 PRINT "VERIF=";VERIF
70 IF VERIF THEN PRINT "LES DEUX,BRAVO"
80 IF NOT (A=2 OR B=3) THEN PRINT "RIEN"
90 IF (A=2 XOR B=3) THEN PRINT "UN SEUL"
100 GOTO 30
```

La ligne 50 est super importante : une condition logique prend une valeur numérique, zéro si elle est fausse et -1 (et non pas 1) si elle est vraie). Dans ce cas simple, les parenthèses ne sont pas obligatoires, c'est pour la clarté.

En ligne 70, on écrit seulement IF VERIF plus court que IF VERIF = -1 ou IF VERIF $<>0$.

En ligne 80, la fonction NOT transforme un 0 en -1 et -1 en 0.

En ligne 90, XOR signifie OU EXCLUSIF, n'accepte *qu'une seule* des deux égalités.

N'utilisez pas la fonction ELSE sous le simple prétexte d'éviter de faire une nouvelle ligne. En effet, ELSE ressemble un peu à un OU EXCLUSIF ; il signifie "Vérifier ce qui me suit *au cas où* la condition qui me précède aurait échoué". Donc, si le premier IF donne oui (ou -1), le ELSE et ce qui le suit ne sont même pas lus. On peut mettre plusieurs ELSE sur une même ligne.

Nous avons insisté lourdement sur le comportement réel des fonctions logiques car c'est leur mauvaise compréhension qui provoque les 90 % de ces fameux bugs fantômes.

Les parenthèses en logique sont comparables à celles dans les formules arithmétiques complexes : n'hésitez pas à en mettre, même si elles paraissent inutiles ; c'est plus clair et plus sûr. On peut, là aussi, imbriquer des parenthèses.

LOCALISONS LA ZONE DE L'ERREUR

Lorsqu'il y a message d'erreur avec numéro de ligne, le coupable est soit sur cette ligne soit en *amont.* Il est alors facile de remonter le courant jusqu'au bug.

En revanche, lorsque le fonctionnement est défectueux mais sans message d'erreur, il faut localiser la zone où il se cache.

La méthode la plus élégante consiste à interrompre le programme çà et là par une ligne dont le numéro se termine par 9 (= ligne provisoire) et comportant la commande STOP. Exemple : 459 STOP.

Après lancement, le programme s'arrêtera sur chaque STOP avec le message ''Break in 459''. Vous ''avez la main'' pour le questionner en mode direct afin de connaître certaines variables en cours ; du genre PRINT J, même faire LIST. Si c'est OK, tapez CONT, et le programme se poursuit jusqu'au prochain STOP et ainsi de suite.

Deux interdits :

— Ne pas modifier une ligne du programme, même très légèrement ; vous perdriez toutes les variables en cours.
— Même effet si vous redémarrez par un RUN, genre RUN 460.

Par contre, vous avez la possibilité de changer la valeur d'une variable ; par exemple en tapant J = 15:CONT et Enter.

Toujours en mode direct, vous pouvrez aussi visualiser le contenu d'un DIM par une boucle FOR NEXT, mais attention au nom de la variable utilisée pour cette boucle...

Le problème est épineux, vous modifiez le programme à plusieurs reprises que vous relancez par RUN, et vous trouvez fastidieux de retaper les mêmes questions après le STOP. Programmez-les :

```
459 PRINT NB,A$,I:STOP
```

DEBUSQUONS LE BUG

Vous avez localisé ce groupe de lignes où se produit le bug, mais c'est un enchevêtrement de IF THEN GOSUB... ELSE GOTO... Par où passe le programme ? Alors, on lâche les chiens, TRON et TROFF (= TRACE ON et TRACE OFF).

Au début de la zone, programmez TRON sur une ligne en 9, et TROFF en fin de zone, aussi sur une ligne en 9. Puis RUN.

Les numéros de lignes sur lesquels ''passe'' le programme apparaissent à l'écran entre crochets. Vous avez donc son itinéraire réel.

NOTE : Eliminez provisoirement de la zone les CLS qui s'y trouveraient...

Bugs du troisième degré : TRON ne vous a rien appris de surprenant car certaines lignes sont très complexes, plusieurs clauses, plusieurs ELSE. Alors on pose les pièges, les FLAGS : après chaque THEN et chaque ELSE, on inserre :

:F1 = 1 ou :F2 = F2 + 1: etc

Après un STOP, il suffira de questionner ces flags. L'auteur a déjà rencontré de redoutables bugs sauvages, mais pas un seul qui ait résisté à ces petits mouchards...

Le bug neutralisé, n'oubliez pas de faire le ménage en supprimant toutes les lignes dont les numéros se terminent par neuf, très faciles à repérer ; ainsi que les flags de piégeages s'il y en a eu.

LES ERREURS DU PASSE

Il s'agit de lignes ''mortes'', désaffectées, obsolètes, recopiées ailleurs, que l'on a oublié d'effacer lorsqu'on triturait le programme pour sa mise au point. Ce genre d'étourderie est chose banale dans un listing très long.

Souvent ces lignes mortes ne gênent absolument pas le bon déroulement du programme, soit parce qu'elles ne sont jamais lues ou parce qu'elles refont une chose déjà faite.

On les découvre généralement grâce au hasard en se plongeant dans le listing. C'est ainsi que nous avons découvert un groupe de trois lignes mortes dans un programme qui fonctionne quotidiennement depuis un an et demi...

Elles ont bien sûr été effacées sur le champ, mais on ne l'a pas avoué au client (ne le répétez pas !).

Si vous n'êtes pas très sûr que la ligne que vous avez découverte soit obsolète, insérez un « ' » après le numéro pour en faire un REM, et relancez le programme.

LES ERREURS DU FUTUR

Un programme en RAM peut êtra abimé accidentellement, par exemple en ''ayant la main'' si on tape 90 et Enter, on a effacé la ligne 90, sans s'en douter. Autre cause (plus rare) : cela vient d'une cassette défectueuse ou d'une version périmée de ce programme (Ah !... si vous aviez inscrit la date après le titre dans le REM de la ligne n° 10...).

Un bon test est de demander sa taille en octets. Pour cela faire :

PRINT 43533 – FREE(" ")

Mais à la condition que le programme n'ait pas été lancé (réservations des DIM, etc.). Si c'est le cas, faites d'abord ''ESC'' puis :

ECRASE:CLEAR:END

(END ferme tous les fichiers ouverts.)

Pour connaître la taille exacte d'un programme sur cassette, faites d'abord une ''initialisation système'' par SHIFT + CTRL + ESC, puis chargez le programme non pas par CTRL + ENTER (= RUN'' ''), mais en tapant :

LOAD'' ''

Il y a alors charge sans RUN et vous pourrez sans risque d'erreur questionner la RAM. Notez bien cette taille en octets sur le listing et sur l'étiquette de la cassette (au crayon sur cette dernière). Cette valeur de référence est très précieuse en cas de doutes.

Chapitre XIV

LE GRAPHISME SIMPLE

Navré de vous décevoir, mais n'espérez pas pouvoir programmer en Basic des dessins animés rapides, comme ceux de certaines cassettes de jeux. En Basic, on peut tout dessiner, mais c'est *lent.* Pour la rapidité, il faut programmer en langage machine, disons plutôt en ''assembleur''. Il n'est pas question d'apprendre cette pratique en un chapitre, alors qu'un gros livre y suffirait à peine.

La conception rationnelle d'un programme graphique (donc sans s'enliser pendant des heures), n'a en fait que peu de rapport avec le Basic général, c'est un domaine un peu à part. Pour y devenir efficace, voici quelques recommandations importantes :

- concevez d'abord sur du papier quadrillé,
- commencez par des dessins simples,
- vous n'apprendrez rien en recopiant des listings très élaborés,
- utilisez surtout les commandes *relatives,* à savoir DRAWR, MOVER et PLOTR,
- mélangez des caractères graphiques pour les petits motifs et les animations,
- ne vous obstinez pas sur une idée si elle s'annonce trop difficile ; modifiez votre projet de dessin.

Ce dernier conseil peut surprendre. En effet, si certains dessins super fastidieux avec un crayon deviennent presqu'enfantins avec un micro, l'inverse est hélas plus fréquent : vous savez écrire un f minuscule (pas d'imprimerie) sur du papier en une fraction de seconde, mais sur un écran... Bon courage ! En somme, il faut composer son dessin en fonction de ce que notre Basic sait faire. Consacrer vingt lignes de programme pour un tracé d'apparence banale est une entreprise absurde, il y a bien mieux à faire.

LES ATOUTS ET LES POINTS FAIBLES

Commençons par les points forts de l'AMSTRAD en graphisme :
- nombreuses fonctions Basic,
- on peut mélanger traits et caractères (PLOT et LOCATE),
- grand nombre de caractères graphiques,
- la très haute résolution du MODE 2,
- l'origine des axes est en bas à gauche et non en haut à gauche comme sur la plupart des micro-ordinateurs,
- le tracé peut ''sortir'' et réintégrer l'écran sans plantage,
- un graphisme a la même dimension dans les trois modes,
- le point d'origine 0,0 étant déplaçable, on a droit aux coordonnées négatives,
- même un trait oblique peut être en plusieurs couleurs,
- tracés simultanés par la fonction EVERY.

LES POINTS FAIBLES
- pas de fonction CIRCLE,
- pas de fonction PAINT,
- quatre couleurs en MODE 1, alors qu'avec 16 couleurs en MODE 0 le tracé est souvent grossier,
- mémoire d'écran trop complexe pour faire des tracés par POKE une pratique courante,
- pas de ''sprites''.

Sachant cela, définissons notre stratégie :

- évitons les grandes courbes et les grands cercles,
- lignes droites et brisées sans problèmes,
- petits éléments graphiques accollés (par PRINT) pouvant donc se mouvoir rapidement, et auxquels peut s'ajouter la ''transparence'',
- les zones de couleurs auront de préférence des limites verticales et horizontales, sauf pour les caractères graphiques pleins (par PEN).

LA CONCEPTION SUR PAPIER QUADRILLE

Prendre du papier quadrillé 5 × 5 mm. Comme les coordonnées d'écran sont de 640 par 400, on a le choix de tracer un rectangle de 16 × 10 cm (avec 1 carreau = 20 points) ou de 32 × 20 cm (où 1 carreau = 10 points).

Puis vous tracez votre dessin *au crayon* (la gomme est à portée de main...). Vous pouvez alors chiffrer vos PLOT et vos DRAW sans calculer, mais en mesurant, d'où moins de risques d'erreur.

En fait, nous vous conseillons vivement de partir d'un point localisé par un PLOT et, à partir de là, partir en coordonnées relatives (DRAWR et MOVER). Le principal avantage est de pouvoir recentrer votre dessin en modifiant *uniquement* les coordonnées du premier PLOT, le reste suit...

Pour mêler des caractères graphiques, rappelez-vous que 1 point de LOCATE vaut 16 points de graphique.

UNE ANIMATION TRES SIMPLE

```
1 'CHAPITRE 14A
10 '  MONTEE D'ESCALIER
20 MODE 1:ORIGIN 0,0:PLOT 185,3
30 FOR I=1 TO 29
40 DRAWR 0,8:DRAWR 8,0:NEXT
50 DRAWR 50,0:DRAWR 0,-240
60 FOR I=1 TO 10
70 LOCATE I,25:PRINT " ";CHR$(250)
80 GOSUB 1000:NEXT
90 FOR P=24 TO 10 STEP-1:GOSUB 1000
100 LOCATE 11+H,P+1:PRINT " "
110 LOCATE 12+H,P:PRINT CHR$(250)
120 H=H+1:NEXT
130 FOR I=1 TO 3:GOSUB 1000
140 LOCATE 25+I,10:PRINT " ";CHR$(250):N
EXT
150 END
1000 FOR A=1 TO 200:NEXT
1010 RETURN
```

Il s'agit d'un petit personnage, CHR$(250), qui monte un grand escalier. Analysons quelques détails.

La ligne 20 initialise tout : tout d'abord ORIGIN 0,0 est une sage précaution au cas où un programme antérieur aurait laissé une autre ORIGIN en mémoire. Enfin, le fameux PLOT de départ. Par défaut, le tracé se fait toujours en INK 2 (ciel).

Les lignes 30 à 50, et elles sont courtes, suffisent à dessiner les marches d'escalier, le pallier et le mur. Alors que si on avait utilisé DRAW au lieu de DRAWR...

En 70, arrivée du personnage. Pour ralentir le mouvement d'une boucle d'attente en GOSUB 1000.

Lignes 90 à 120, c'est la montée en diagonale du personnage avec deux LOCATE paramétrés, dont un pour effacer la position précédente (ligne 100).

Lignes 130 et 140, il fait trois pas sur le palier et s'arrête.

Vous remarquerez que le tracé de l'escalier est ultra rapide sur l'écran, une fraction de seconde malgré 61 instructions DRAWR...

TRACES RAPIDES DE CERCLES

Vous avez sans doute été déçu par la lenteur du tracé d'un cercle par la méthode du manuel, qui consiste à faire 360 PLOT de degré en degré... Et on perçoit le pointillage en MODE 2 ! Nous vous offrons un petit cadeau : un tracé continu, un cercle parfait (même en MODE 2), qui ne dure que 1,5 seconde, quel que soit le diamètre.

```
2000 ' CERCLES en 1.5 seconde
2010 CLS
2020 INPUT "CENTRE:X";X
2030 INPUT "         Y";Y
2040 INPUT "       INK";COL:COL=COL-1
2050 INPUT "    RAYON";R
2060 CLS
2070 GOSUB 54000
2080 GOTO 2050
54000 '   TRACE DE CERCLE
54010 DEG:PLOT X+R,Y,1+COL
54020 FOR A%=0 TO 360 STEP 10
54030 DRAW R*COS(A%)+X,R*SIN(A%)+Y:NEXT
54040 RETURN
```

Tout le tracé est dans le très court sous-programme 54000 (un de plus sur la cassette UTIL). Dans la pratique future, il suffira de programmer :

```
R= :X= :Y= :COL= :GOSUB 5400
```

La variable COL est facultative, c'est le numéro d'INK désiré. Par défaut (COL = 0), le tracé se fera en INK 1 (et non pas 2) ; cette valeur de 1 par défaut est une sécurité pour le MODE 2.

Dans ce module, nous voyons que le tracé se fait par DRAW au pas de 10°, soit 360/10 = 36 cordes de 10°. Pour vous donner une idée de qualité, passez en MODE 2 avec d'abord :

```
X=320 : Y=200 : R = 199
```

Puis diverses valeurs de R : 150, 100, 60, 10, 5.

Si vous voulez avoir le centre, commandez :

```
PLOT X,Y,1:GOSUB 54000
```

*NOTE :*Pour tracer un grand cercle par des PLOT au pas de un degré, il faut 13 secondes ; pour faire disparaître le tracé en pointillés, il faut un STEP de 0.5°, d'où une durée de 26 secondes... Nous sommes donc 17 fois plus rapides.

Rappelons que l'on peut modifier la couleur en fonction de l'angle, par exemple dans la ligne 54030, insérez après Y :

```
,A%/25+1
```

A lancer en MODE 0 avec COL = 1.

Rappelons également que l'AMSTRAD tolère les débordements d'écran graphique. Relancez le programme avec X = 320:Y = 0:R = 320.

UN PETIT DESSIN ANIME

Nous voulons vous montrer qu'il est facile et amusant de concevoir une animation en Basic. Ce petit scénario de 31 secondes (1526 octets) n'a pourtant rien à envier à certaines cassettes du commerce, sur le plan du graphisme. Nous n'avons pas voulu ''tricher'' en définissant des caractères graphiques, mais en utilisant uniquement ceux de l'AMSTRAD appelés bien sûr en PRINT CHR$( ).

Aucune astuce nouvelle de programation, c'est tout du déjà vu, donc aucune ligne ne sera pour vous un mystère. Des REM indiquent les diverses séquences. Nous n'avons pas ajouté le son afin de ne pas surcharger.

Le rôle de ce programme est avant tout un tremplin : après l'avoir tapé, nous vous invitons à le modifier, à le compléter, voire même à continuer l'histoire.

Sur le sol de Vénus, aux couleurs bizarres, arrivent deux petits Martiens verts, ils attendent. Arrive la soucoupe martienne qui se pose.

Mais au moment d'embarquer, un Vénusien rouge survient et tue un Martien. L'autre se retourne, abat le ''méchant'' et monte seul dans la soucoupe. Flammes de réacteurs et décollage à la verticale.

NOTE : Avant de taper ce programme, chargez le programme KEY ; les touches de fonction vont être très utiles !

```
10 '  RIFIFI SUR VENUS
20 'COPYRIGHT ARCHAMBAULT 1985
30 MODE 0:BORDER 1:PEN 1
40 LOCATE 3,3:PRINT "RIFIFI sur VENUS":F
OR I=1 TO 3000:NEXT
50 LOCATE 3,3:PRINT CHR$(18)
60 A$=" "+CHR$(222)+CHR$(207)+CHR$(223)+
" "
70 B$=CHR$(208)+CHR$(213)+CHR$(210)+CHR$
(212)+CHR$(208)
80 P$=CHR$(250)+" "+CHR$(250)
90 '  DECOR
100 FOR I=1 TO 15
110 PLOT 0,I,I:DRAWR 640,0:NEXT
120 FOR H=1 TO 11:PEN 12:LOCATE H,24:PRI
NT " ";P$:FOR I=1 TO 200:NEXT:NEXT
130 FOR I=1 TO 3000:NEXT
140 '  ARRIVEE DE LA SOUCOUPE
150 FOR I=1 TO 13
160 LOCATE I,10:PEN 2:PRINT " ";A$
170 LOCATE I,11:PEN 3:PRINT " ";B$
180 FOR J=1 TO 50:NEXT
190 NEXT
200 FOR I=1 TO 2000:NEXT
210 FOR V=10 TO 21
220 LOCATE 14,V:PRINT SPC(6)
230 LOCATE 14,V+1:PEN 2:PRINT A$
240 LOCATE 14,V+2:PEN 3:PRINT B$
250 FOR I=1 TO 30:NEXT:NEXT
260 LOCATE 15,24:PEN 1:PRINT CHR$(47);"
";CHR$(92)
270 ' ARRIVEE DU MECHANT
```

```
280 LOCATE 1,24:PEN 3:PRINT CHR$(250):FO
R I=1 TO 2000:NEXT
290 PLOT 30,25,3:DRAWR 335,0:FOR I=1 TO
500:NEXT:DRAWR -335,0,0
300 LOCATE 12,24:PRINT CHR$(210)
310 FOR I=1 TO 1000:NEXT
320 '  LA RIPOSTE
330 LOCATE 14,24:PEN 12:PRINT CHR$(251)
340 FOR I=1 TO 1000:NEXT
350 PLOT 416,25,12:DRAWR -400,0:FOR I=1
TO 500:NEXT:DRAWR 400,0,0
360 LOCATE 1,24:PEN 11:PEN 11:PRINT CHR$
(244)
370 FOR I=1 TO 2000:NEXT
380 '  L'EMBARQUEMENT
390 FOR H=13 TO 14
400 LOCATE H,24:PEN 12:PRINT " ";CHR$(25
0):FOR I=1 TO 400:NEXT:NEXT
410 LOCATE 15,24:PEN 1:PRINT CHR$(47):FO
R I=1 TO 2000:NEXT
420 SPEED INK 3,3
430 LOCATE 15,24:PEN 14:PRINT " ";CHR$(2
07);" "
440 FOR I=1 TO 3000:NEXT:LOCATE 16,24:PR
INT " "
450 FOR V=21 TO 2 STEP-1
460 LOCATE 14,V:PEN 2:PRINT A$
470 LOCATE 14,V+1:PEN 3:PRINT B$
480 LOCATE 14,V+2:PRINT SPC(6)
490 FOR I=1 TO 30:NEXT:NEXT
500 LOCATE 7,9:PRINT "MORALITE:"
510 LOCATE 4,12:PEN 12:PRINT "1 VERT,2 V
ERTS,"
520 LOCATE 2,14:PRINT "BONJOUR LES DEGAT
S"
530 SPEED INK 20,20:PEN 1:LOCATE 1,1:PRI
NT
540 END
550 '-----FIN DE LISTING------
```

Remarquez la ligne 530 qui restitue les options par défaut.

LES SURFACES A BORDS OBLIQUES

Comme nous l'avions dit, il n'y a rien de prévu dans le Basic pour colorer des surfaces à bords obliques, mais c'est néanmoins faisable. La technique consiste à tracer une suite de traits presque jointifs et parallèles. En MODE 0, un STEP de 4 (en X ou en Y) est suffisant pour obtenir une surface unie. En revanche, en MODE 1, il faut un STEP de 2.

Le programme qui suit donne une image fixe représentant une maison en perspective, avec un mur à l'ombre, deux fenêtres et une porte. Elle est dans un pré avec, en arrière-plan, la mer, une grande île, et un ciel d'été avec deux mouettes. Le tout en 720 octets seulement (REM compris) ; le tracé s'effectue en 4,8 secondes.

```
600 '  MAISON DE CAMPAGNE
610 MODE 0:BORDER 13:L$=SPACE$(20)
620 '  DECOR
630 FOR I=1 TO 17:PAPER 2:LOCATE 1,I:PRI
NT L$:NEXT
640 PAPER 6:LOCATE 1,18:PRINT L$
650 FOR I=19 TO 25:PAPER 12:LOCATE 1,I:P
RINT L$:NEXT
660 '  ILE
670 FOR I=1 TO 240 STEP 4:PLOT 400+I,144
,0:DRAWR 125,50:NEXT
680 '  TOITURE
690 FOR I=100 TO 200 STEP 4
700 PLOT I,100:DRAWR 50,50,3:NEXT
710 DRAWR 50,-50:MOVER 4,0:DRAWR -50,50
720 '  MURS
730 FOR I=110 TO 200 STEP 4
740 PLOT I,50:DRAWR 0,50,1:NEXT
750 FOR I=200 TO 250 STEP 4
760 PLOT I,50,9:DRAWR 0,50+I-200:NEXT
770 FOR I=250 TO 295 STEP 2
780 PLOT I,50,9:DRAWR 0,99-I+250:NEXT
790 '  FENETRES ET PORTE
800 PEN 0:PAPER 1:LOCATE 5,21:PRINT CHR$
(233);CHR$(233)
```

```
810 PLOT 260,50,0:DRAWR 0,40:DRAWR 17,0:
DRAWR 0,-38:DRAWR -17,0
820 '  OISEAUX
830 PAPER 2:LOCATE 15,5:PRINT CHR$(198):
LOCATE 17,3:PRINT CHR$(198)
840 PAPER 0:PEN 1:LOCATE 1,1:PRINT
850 END
860 '----- FIN DE LISTING -----
```

Pour le décor (ciel, mer et pré), on utilise des bandes de PAPER en LOCATE parce que, par la suite, c'est plus souple que des fonds en WINDOW.

L'île, le toit et les murs sont des DRAWR jointifs. La difficulté est le calcul des progressions horizontales par PLOT paramétrés.

Par la hauteur du mur à l'ombre, la longueur des DRAWR verticaux est aussi paramétrée (en deux parties).

Les fenêtres et les oiseaux sont des caractères graphiques, la porte est en DRAWR.

CONCLUSION : Ce n'est pas tout simple, mais c'est faisable et le résultat est comme on dit ''payant''.

LES DISQUES PLEINS

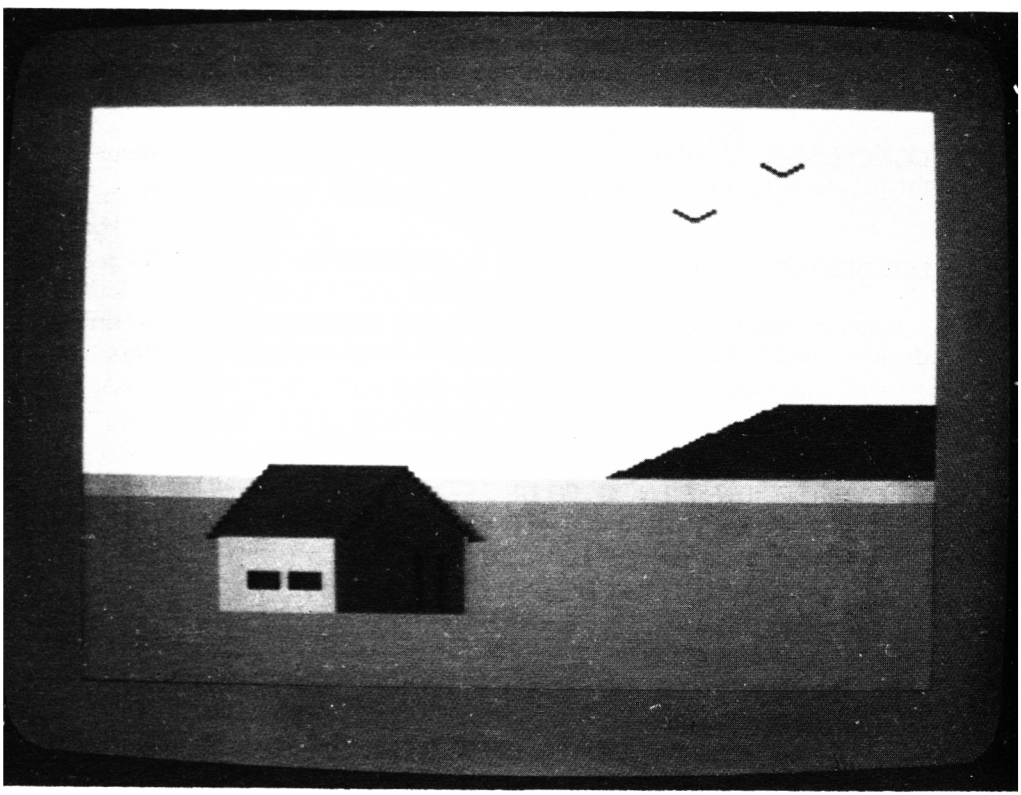
Nous savons tracer un cercle (rapidement), mais comment tracer un disque ''plein'' d'une couleur déterminée ? Deux méthodes au choix :

```
1 'CHAPITRE 14C
3000 ' DISQUE PLEIN TRACE VERTICAL
3010 ORIGIN 0,0:DEG
3020 X=320:Y=200:R=199:'CENTRE & RAYON
3030 FOR A=0 TO 180 STEP 0.5
3040 PLOT X+COS(A)*R,Y+SIN(A)*R,2
3050 DRAWR 0,-2*R*SIN(A)
3060 NEXT
3100 ' DISQUE PLEIN TRACE HORIZONTAL
3110 R=100
3120 FOR A=90 TO 270 STEP 1
3130 PLOT X+COS(A)*R,Y+SIN(A)*R,3
3140 DRAWR -2*R*COS(A),0
3150 NEXT
3160 END
```

Nous obtenons d'abord un grand disque bleu-ciel, puis un disque rouge plus petit et concentrique. Pour obtenir un fragment de cercle, on optera pour l'une ou l'autre méthode selon l'orientation de ce fragment. Remarquez que l'on peut augmenter le STEP quand le rayon diminue.

Pour faire un croissant de Lune, faire un disque blanc, décaler le centre (X) à gauche ou à droite, puis dessiner un disque couleur PAPER de rayon légèrement supérieur.

Vous voyez qu'en réfléchissant un peu, on arrive à (presque) tout faire...



Chapitre XV

LA MÉMOIRE D'ÉCRAN

Les notions exposées dans ce chapitre ne sont pas du tout indispensables pour devenir un bon progammeur en Basic ; elles permettent d'aller plus loin en triturant la zone mémoire de l'écran par des PEEK ou des POKE. Pour faire quoi ? Tout d'abord faire des programmes de ''HARD COPY'' qui exécutent sur imprimante une sorte de photocopie de l'écran en cours, surtout utile pour les graphismes (vous pourrez utiliser ces programmes même si vous ne comprenez pas leurs fonctionnements). Avec ces notions, et de la patience, vous pourrez aussi créer des effets colorés inaccessibles par les seules fonctions Basic.

Ce chapitre est assez ''difficile'', c'est le seul de cet ouvrage ; si vous êtes plutôt débutant, *n'hésitez pas à le sauter* car il est totalement indépendant des autres.

GENERALITES SUR LA MEMOIRE

Le CPC 464, comme la plupart des micro-ordinateurs de moins de 20 000 FF, utilise un microprocesseur ''8 bits'' (un octet). Pour aller chercher un octet en mémoire, il faut que ce dernier ait une adresse. Un octet peut prendre 256 valeurs possibles ($256 = 2^8$). Pour les adresses, disons un octet pour les ''colonnes'' et un octet pour les ''lignes'' d'un ''tableau d'adresses'', donc $256 \times 256 = 65536$ ''cases mémoire'', qu'on numérote de 0 à 65535. C'est ce qu'on appelle 64 k-octets car le ''kilo-octet'' vaut en réalité 1024 octets... Donc 64 k-o est le maximum pour un micro 8 bits.

Il faut les partager entre la ROM (le Basic résident) et la RAM (programmes + variables en cours + les points lumineux envoyés à l'écran). Avec un Basic de 32 k-o, il ne resterait que 32 k-o pour la RAM, d'où vingt et quelques k-o disponibles pour programmes plus fichiers, c'est ce qui se passe avec les MSX et bien d'autres. Nous, nous avons 32 k-o de Basic, 43 k-o disponibles + 16 k-o réservés à l'écran ! La raison de ce ''miracle'' est que l'AMSTRAD ''recharge et décharge'' la portion de la ROM dont il a besoin (c'est une vue d'esprit). De ce fait, une large partie de la RAM est plutôt ''mouvante'', ce qui va donner bien du plaisir aux amateurs de PEEK et de POKE...

La partie mémoire écran occupe une zone fixe, de 49152 à 65487, mais le fait que l'on puisse mélanger texte et graphisme, et ce en trois MODES, fait que sa structure est d'une extrême complexité (même principe que sur les APPLE II, mais en pire). C'est ce qui explique que l'affichage des caractères est assez lent, exemple, quand on fait LIST sur un long programme.

Dans le manuel, les adresses sont indiquées en hexadécimal (préfixe & et quatre caractères). Il est pratique de savoir faire les conversions hexadécimal-décimal dans les deux sens (Note : le préfixe hexadéci ne signifie pas ''six'' mais ''seize'').

LES CONVERSIONS HEXA/DECIMAL/BINAIRE

En binaire, on a deux symboles 1 et 0, en décimal on a dix symboles 0 à 9, en hexa on en a seize, 0 à 9 + A, B, C, D, E et F (&F = 15;&A = 10).

On représente la valeur d'un octet par deux symboles hexa, ainsi & FA = (256 × 15) + 10 = 250. Il suffit de taper PRINT &FA et 250 apparaît à l'écran. De même, &FF = 255, c'est bien le maximum pour un octet.

Une adresse tient sur deux octets, exemple &AFC8. Vous ne pouvez pas faire PRINT &AFC8 car l'AMSTRAD ne sait pas faire la conversion sur plus d'un octet, vous êtes obligé de décomposer :

```
PRINT &AF*256+&C8
```

et la réponse apparaît : 45000.

L'opération inverse est beaucoup plus facile :

```
PRINT HEX$(45000)
```

l'écran répond AFC8 (le & est sous-entendu).

Pour traduire une longue suite de valeurs hexa en décimal, vous gagnerez du temps en utilisant ce très court programme :

```
1 'CHAPITRE 15A
10 '  CONVERSION HEXA --> DECIMAL
20 INPUT "&",H$
30 H$=RIGHT$("0000"+H$,4)
40 G$="&"+LEFT$(H$,2):D$="&"+RIGHT$(H$,2
)
50 PRINT VAL(G$)*256+VAL(D$)
60 PRINT:GOTO 20
70 END
```

Passons à la représentation binaire d'un nombre. Tapez, par exemple : PRINT BIN$(45956), vous obtenez une suite de 16 bits 1 et 0. Essayez maintenant PRINT BIN$(3), vous obtenez ''11'' (c'est une chaîne) ; avec PRINT BIN$(3,16), vous avez 14 zéros suivis de ''11''.

Gare à l'opération inverse par &X, elle traduit cette ''image binaire'' en un nombre dit entier, c'est-à-dire compris entre −32768 et +32767 ! Un zéro suivi de quinze 1 donne 32767, tandis que 1 suivi de quinze zéros donne −32768.

Heureusement, vous n'avez pas à vous en préoccuper car dans la pratique les seules images binaires qui nous intéressent sont *sur un octet,* donc de 1 à 255.

LE PLAN MEMOIRE DE L'ECRAN

Nous y voilà !

Deux cent lignes horizontales de 80 octets chacune. Dans une ligne, les numéros d'octets se suivent de gauche à droite. Prenons le cas le plus simple, celui du MODE 2 : chaque bit représente un ''pixel'' ou point à l'écran, ainsi 255 fait un tiret continu (huit bits 1). La ligne en haut de l'écran va de 49152 à 49231, soit 80 × 8 = 640 points (ce qui explique qu'en coordonnées graphiques l'échelle horizontale aille de 1 à 640).

Hélas l'octet suivant , le 49232, n'est pas au départ de la deuxième ligne mais huit lignes plus bas ! Et ainsi de suite au pas de 8 lignes. Après un premier ''passage'' de 25 lignes espacées de 8, commence le deuxième passages sur les lignes du dessous. Pour tout comprendre, essayez ceci :

```
100 ' BALAYAGE DE L'ECRAN
110 MODE 2
120 FOR A=49152 TO 65487
130 POKE A,3
140 NEXT
150 END
```

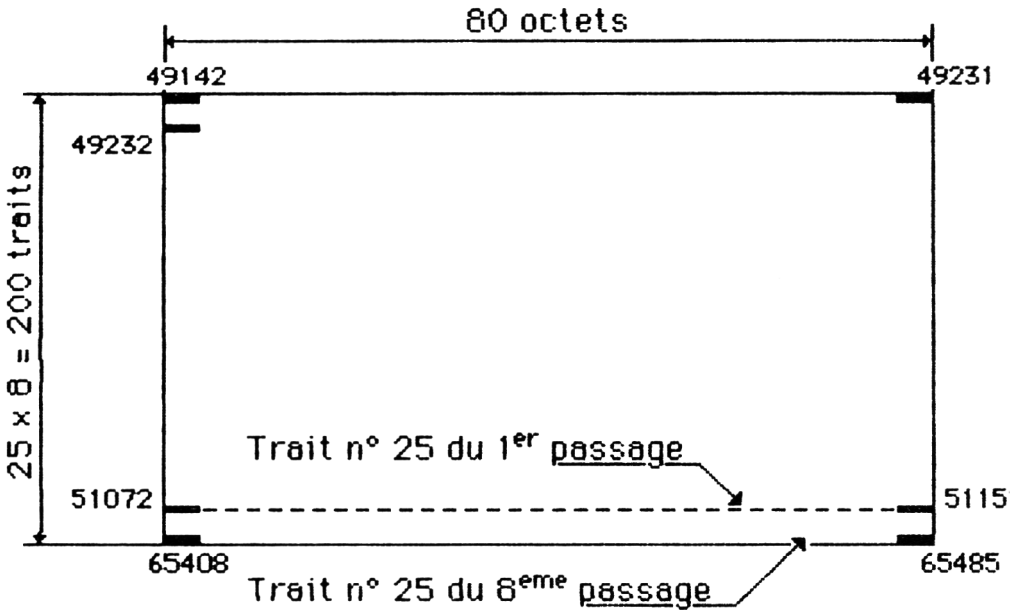


Figure XV 1
La mémoire d'écran est constituée de 200 ''traits'' de 80 octets chacun, mais la continuité des traits est au pas de huit.

				Passage n°	Trait n°
14336	49152	49153	49154	1	1
	51200	51201	- - - -	2	
2048	53248	- - - -	- - - -	3	
	55298	- - - -	- - - -	4	
	57344	- - - -	- - - -	5	
	59392	- - - -	- - - -	6	
	51440	- - - -	- - - -	7	
	63488	- - - -	- - - -	8	
	49232	49233	- - - -	1	2
	51280	- - - -	- - - -	2	

Figure XV 2
La mémoire de l'angle supérieur gauche de l'écran ''vue à la loupe''.

Un joli store vénition !

Les figures XV 1 et XV 2 vous donnent le plan des adresses d'écran. Vous constatez qu'entre deux ''tirets'' (deux octets) situés à l'écran, l'un en-dessous de l'autre, il y a un écart de 2048 octets (= 2 k-octets), qu'entre deux lignes d'un même ''passage'' (donc espacées de 8 lignes d'écran), il y a un écart de 14336 octets.

A présent, modifiez la ligne 110 en mettant MODE 1. Surprise ! Les traits sont rouges (= PEN 3). En ligne 110, mettez MODE 0 : les traits clignotent en rose/bleu ciel (= PEN 15). Nous abordons alors le code des couleurs en fonction du MODE.

LE CODAGE DES COULEURS

En MODE 2, rien de plus simple : comme il n'y a que deux couleurs possibles, un bit 1 fait un pixel PEN, un bit 0 un pixel PAPER (Pixel est la contraction anglaise de PICTURAL ELEMENT).

Lancez ce programme qui va dessiner en 0 et 1 les images binaires des quatre premiers caractères sur un écran en MODE 2.

```
200 '  IMAGE BINAIRE D'UN CARACTERE
210 MODE 2
220 CLS:PRINT "aV8/"
230 FOR A=49152 TO 63491 STEP 2048
240 FOR N=0 TO 3
250 CB=PEEK(A+N)
260 PRINT BIN$(CB,8);".";
270 NEXT:PRINT CHR$(13)
280 NEXT
290 END
```

Le résultat est spectaculaire. On y retrouve les compositions illustrées dans le manuel AMSTRAD page A3.2 et suivantes.

A présent, modifions la ligne 210 afin d'avoir MODE 1 (ou faites MODE 1:RUN 220). Oh ! Que c'est vilain ! Sur chaque ''pavé'', on n'a qu'une moitié de chaque caractère. Il faudrait accoller les moitiés gauches des pavés pour reconstituer le texte. Les moitiés droites des pavés (ou des octets) ne comportent que des 0. Ces zéros constituent en fait le code couleur de PEN (PEN 1 en ce cas) : apparions – les gauche + droite, on a '01'' quand il y a quelque chose à tracer, or &X01 = 1 (PEN 1).

Tapons PEN 2 et, toujours en MODE 1, faisons RUN 220 : les 1 sont à droite et les zéros à gauche, &X10=2 (PEN 2).

Tapons PEN 3 et même manœuvre. Dans chaque pavé, le demi caractère est répété à gauche et à droite, or &X11=3.

Puisque dans chaque octet on associe un bit de la moitié droite à un de la moitié gauche, il n'y a que quatre combinaisons possibles :

"00" (=0) ; "01" (=1) ; "10" (=2) et "11" (=3)

C'est pourquoi on n'a que quatre couleurs en MODE 1.

Et que se passe-t-il sur l'écran vidéo ? Sur chaque image binaire d'un octet (le "tiret"), chaque bit 1 est *doublé*, et dans la couleur du PEN. Un caractère occupe donc deux pavés, et comme il y a 80 octets par ligne, cela fait bien 40 caractères par ligne d'écran.

En MODE 0, c'est le même principe, mais en divisant par deux, un quart de caractère par pavé. Pour apparier les bits, on a alors 16 combinaisons possibles (les seize couleurs). A l'écran, chaque bit 1 est quadruplé dans la même couleur.

LE HARD COPY D'ECRAN

Il faut que votre imprimante possède l'instruction "BIT IMAGE", c'est le cas de la plupart des imprimantes "à aiguilles" ou "matricielles". En usage normal, une imprimante traduit l'octet reçu en dessinant le caractère ayant ce code ASCII, ici c'est beaucoup plus simpliste. Les huit aiguilles de la tête d'impression illustrent l'image binaire de cet octet. Ces dessins de bits vont être des "barres" verticales, or sur l'écran ils sont horizontaux ! Il suffit de tourner notre graphisme d'un quart de tour sur le papier. L'imprimante commencera par le bord vertical gauche en terminant par le bord droit de l'écran ; le bas de l'écran sera donc le long de la marge gauche du papier. Ce n'est pas gênant.

L'auteur utilise une EPSON RX80 (les modèles FX ont les mêmes codes). Si la vôtre est d'une autre marque, il va falloir modifier les libellés des codes de commandes ; nous légenderons les nôtres afin que vous puissiez les traduire pour votre machine. Si vous trouvez les mêmes, ne soyez pas surpris car une bonne quinzaine de "marques" sont des EPSON rebaptisées.

Il y a deux programmes différents, pour le MODE 2 et le MODE 1 (la version MODE 0 ne satisfait pas l'auteur).

Il s'agit en fait de sous-programmes utilitaires (cassette UTIL...).

```
55000 'HARD COPY AMSTRAD MODE 1
55010 PRINT #8
55020 DIM Z(200)
55030  PRINT #8,CHR$(27);CHR$(64);CHR$(2
7);"3";CHR$(24);
55040 PRINT #8,CHR$(13)
55050 PRINT #8,CHR$(15);
```

```
55060 FOR COL=49152 TO 49231 STEP 2:I=0
55070 FOR LI=COL TO COL+1920 STEP 80
55080 FOR T=LI TO LI+14336 STEP 2048
55090 I=I+1:Z$=LEFT$(BIN$(PEEK(T),8),4)+
   LEFT$(BIN$(PEEK(T+1),8),4)
55100 Z(I)=VAL("&X"+Z$):IF I=200 THEN GO
SUB 55130
55110 NEXT:NEXT:PRINT #8,CHR$(13):NEXT
55120 RETURN
55130 PRINT #8,CHR$(27);"K";CHR$(100);CH
R$(0);
55140 FOR J=200 TO 101 STEP -1
55150 PRINT #8,CHR$(Z(J));:NEXT
55160 PRINT #8,CHR$(27);"K";CHR$(100);CH
R$(0);
55170 FOR J=100 TO 1 STEP -1
55180 PRINT #8,CHR$(Z(J));:NEXT
55190 RETURN
```

Temps d'exécution 4'15''. Format 112 × 84 mm.

```
56000 'HARD COPY MODE 2 AMSTRAD
56010 PRINT #8
56020 DIM Z(200)
56030  PRINT #8,CHR$(27);CHR$(64);CHR$(2
7);"A";CHR$(8);
56040 PRINT #8,CHR$(13)
56050 PRINT #8,CHR$(15);
56060 FOR COL=49152 TO 49231:I=0
56070 FOR LI=COL TO COL+1920 STEP 80
56080 FOR T=LI TO LI+14336 STEP 2048
56090 I=I+1:Z(I)=PEEK(T):IF I=200 THEN G
OSUB 56120
56100 NEXT:NEXT:PRINT #8,CHR$(13):NEXT
56110 RETURN
56120 PRINT #8,CHR$(27);"K";CHR$(100);CH
R$(0);
56130 FOR J=200 TO 151 STEP -1:P=Z(J)
56140 PRINT #8,CHR$(P);CHR$(0);:NEXT
56150 PRINT #8,CHR$(27);"K";CHR$(100);CH
R$(0);
56160 FOR J=150 TO 101 STEP -1:P=Z(J)
56170 PRINT #8,CHR$(P);CHR$(0);:NEXT
```

```
56180 PRINT #8,CHR$(27);"K";CHR$(100);CH
R$(0);
56190 FOR J=100 TO 51 STEP -1:P=Z(J)
56200 PRINT #8,CHR$(P);CHR$(0);:NEXT
56210 PRINT #8,CHR$(27);"K";CHR$(100);CH
R$(0);
56220 FOR J=50 TO 1 STEP -1:P=Z(J)
56230 PRINT #8,CHR$(P);CHR$(0);:NEXT
56240 RETURN
```

Temps d'exécution 7'30''. Format 225 × 170 mm.

LEGENDES DES CODES EPSON

CHR$(27);CHR$(64) = vide le buffer (initialisation)
CHR$(27);''3'';CHR$(24) = interlignes de 24/216 de pouce
CHR$(27;''A'';CHR$(8) = interlignes de 8/72 de pouce
CHR$(15) = caractères condensés (17 par pouce)
CHR$(27);''K'' = mode bit-image

LE PRINCIPE

On explore des bandes verticales de l'écran de largeur 1 octet. Ces 200 octets sont mis en DIM Z puis ils sont retransmis à l'imprimante qui écrit donc une ligne. Passage à la colonne précédente et ainsi de suite. En jouant sur divers paramètres d'impression, nous sommes parvenus à restituer les proportions hauteur × largeur de l'écran (approximativement). En MODE 0, nous ne sommes pas (encore) parvenus à obtenir des dimensions d'image imprimées satisfaisantes, d'où l'absence de la version MODE 0.

Très important : en MODE 1, les inscriptions doivent être en PEN 1 ou en PEN 3 (pas en PEN 2) car nous ne prenons que la moitié gauche des images binaires (ligne 55090).

Suite à un petit bug de la ROM de l'EPSON RX 80, nous n'envoyons en ''bit image'' que des suites de 50 ou 100 octets à la fois.

REMARQUE

C'est aussi pour des raisons de proportions d'écran que les coordonnées graphiques sont de 640 × 400 alors que logiquement ce serait 640 × 200.

CODAGE BINAIRE D'UN CARACTERE

Vous voulez définir un caractère qui se trouve être un caractère existant mais légèrement modifié. Vous trouvez fastidieux de redessiner sa grille 8 × 8 et de totaliser toutes ces lignes. Alors, utilisez ce programme qui vous fait instantanément tout ce travail.

```
300 ' CODAGE BINAIRE D'UN CARACTERE
310 INPUT "CARACTERE OU CODE ASCII";C$:M
ODE 1
320 IF VAL(C$)>32 THEN PRINT CHR$(VAL(C$
));SPC(2);C$:PRINT:GOTO 340
330 PRINT C$;SPC(2);ASC(C$):PRINT
340 FOR A=49152 TO 63488 STEP 2048
350 G$=LEFT$(BIN$(PEEK(A),8),4)
360 D$=LEFT$(BIN$(PEEK(A+1),8),4)
370 C$=G$+D$
380 PRINT C$;SPC(2);VAL("&X"+C$)
390 NEXT
400 PRINT:GOTO 310
410 END
```

Tapez indifféremment le caractère, s'il existe, au clavier ou son code ASCII. L'écran vous soumet sa fiche signalétique complète, à savoir :

Le caractère, son code ASCII, sa configuration binaire en "1" et "0" et les totaux de chaque ligne. Exemple avec X majuscule : Code ASCII 88, codage 198, 108, 56, 56, 108, 198, 198, 0.

Faites un autre essai en tapant 251.

Ce programme refuse certains signes de ponctuation tels que + − , . &. En ce cas entrez le code ASCII.

Chapitre XVI

TEMPS, JOY ET MUSIQUE

Passons à des notions désormais moins sévères, d'autant plus qu'elles sont très utilisées pour les jeux.

LES FONCTIONS TEMPS

La plus ''puissante'' nous semble être la fonction TIME. C'est un compteur qui ne se remet à zéro que par la remise sous tension, soit par l'interrupteur ou par SHIFT + CTRL + ESC. Il compte en 1/300e de seconde. En revanche, il s'arrête momentanément lorsque le magnéto-cassette lit ou enregistre.

Voulez-vous savoir depuis combien d'heures votre AMSTRAD est allumé ? Tapez :

PRINT TIME /300/3600

Un sacré mouchard !

La valeur renvoyée par TIME est un nombre *réel,* alors, même en 300e de seconde, cela représente beaucoup de temps, d'années avant qu'il ne cale... En effet, $300 \times 3600 \times 24 \times 365 = 9.4$ E9 unités par an, ''seulement''.

Les fonctions EVERY (= chaque) et AFTER (= après) utilisent quatre autres chronomètres qui eux, attention, comptent en cinquantièmes de seconde (= 0,02 s). Ces chronomètres sont numérotés 0, 1, 2 et 3, le n° 3 est prioritaire. Eux aussi s'arrêtent quand le magnétophone lit ou enregistre. Autre différence, ils comptent de 0 à 32767, donc un délai maximum programmable de 10,9 minutes. Voici un programme qui transforme le haut de l'écran en pendulette. Il utilise TIME et va nous montrer certains caprices de EVERY.

```
10 '  MONTRE BASIC
20 CLS
30 LOCATE 2,2:PRINT "Entrez l'Heure en s
eparant par des ,"
40 LOCATE 12,4:INPUT "HH,MM,SS";HD,MD,SD
:CLS
50 TD=HD*3600+MD*60+SD
60 T0=TIME/300
70 EVERY 500,3 GOSUB 100
80 WHILE INKEY$="":WEND
90 END
100 ' CALCUL DE L'HEURE
110 LOCATE 1,1:PRINT CHR$(18)
120 DELAI=TIME/300-T0
130 HS=TD+DELAI:'HEURE EN "
140 H=FIX (HS/3600):R=HS-(H*3600)
150 M=FIX (R/60)
160 S=HS-H*3600-M*60:S=INT(S)
170 IF H>24 THEN H=H-24
180 LOCATE 3,1:PRINT "Il est";H;"heures,
";M;"minutes et";S;"s"
190 RETURN
```

La ligne 70, provoque un calcul et un affichage de l'heure toutes les 10 secondes. Si on appuie sur une touche (ligne 80), le programme s'arrête et l'affichage n'est plus ''rafraîchi''. Attendons une minute ou deux puis tapons, en mode direct :

GOSUB 100, et Enter

La ligne d'affichage de l'heure se met à clignoter rapidement. Ce sont tous les EVERY qui n'ont pas été exécutés, ils s'exécutent à présent ! Enfin, l'heure correcte est affichée. Conclusion très importante : même le programme stoppé par END les cycles EVERY continuent et sont *mémorisés.* Comment arrêter le chronomètre d'EVERY ? Deux méthodes : DI et REMAIN. Ajoutons la ligne :

85 DI

et relançons par RUN. Après le premier affichage, pressons une touche, attendons une minute environ et tapons GOSUB 100. Il y a affichage immédiat sans clignotement car DI supprime *toutes* les interruptions, et ce sur les *quatre* chronomètres 0 à 3 (on pourra les remettre en service par la commande EI).

Modifions la ligne 85 :

85 PRINT REMAIN (3)

On relance par RUN, arrêt par touche seulement 3 à 5 secondes après un affichage. Il apparaît un nombre inférieur à 500 ; c'est le nombre de cinquantièmes de seconde qui restaient à attendre pour finir le cycle. D'autre part, le chrono n° 3, et seulement celui-là, est arrêté. Un GOSUB 100 donnera l'heure exacte sans clignotement.

En commandant EVERY 500,3, on a mis le chrono n° 3 à la valeur 500, il décompte et arrive à zéro, action (ici GOSUB 100), il se remet à 500 et ainsi de suite. Le REMAIN (3) a provoqué aussi sa mise à la valeur zéro. C'est le même fonctionnement avec AFTER.

Un des grands points forts de l'AMSTRAD est de pouvoir faire du TEMPS PARTAGE par la commande EVERY, c'est-à-dire faire plusieurs choses alternativement mais à un rythme suffisamment rapide pour que le spectateur ait l'impression qu'elles sont simultanées.

Le programme suivant va tracer lentement deux diagonales, une rouge, une bleu-ciel, qui semblent bien simultanées. Cette possibilité sera en outre très utile pour des programmes de jeux.

Nous en profitons pur illustrer la commande AFTER qui se réserve une action future dans un délai déterminé, et ce indépendamment de ce qui se sera passé entre-temps (sauf un END ou un DI) ; donc à manier avec précaution car un délai mal positionné peut interrompre fâcheusement une action en cours...

```
300 ' ASPECT SIMULTANE PAR EVERY
310 MODE 1
320 PRINT "ON PART DANS DEUX SECONDES"
330 AFTER 100,0 GOSUB 400
340 AFTER 730,3 GOSUB 700
350 WHILE Y<400:WEND
360 LOCATE 7,3:PRINT "ON EFFACE DANS SIX
 SECONDES"
370 WHILE FF=0:WEND
380 DI
390 END
400 ' DEPART
410 PRINT CHR$(7)
420 EVERY 4,1 GOSUB 500
430 EVERY 4,2 GOSUB 600
440 RETURN
```

```
500  ' TRAIT ROUGE
510 PLOT X,Y,3:DRAWR 5,5
520 X=X+5:Y=Y+5
530 RETURN
600  ' TRAIT BLEU CIEL
610 PLOT 640-H,V,2:DRAWR -5,5
620 H=H+5:V=V+5
630 RETURN
700  ' FINAL
710 CLS
720 FF=1  :  '= FLAG DE FIN
730 RETURN
```

Les boucles d'attente des lignes 350 et 370 sont très importantes. Sans ces barrières, le programme irait directement en 380, 390 !

La remise à zéro de la ligne 380 est une bonne habitude à prendre.

LES PAUSES

On a l'habitude du suspendre le déroulement d'un programme par, faute de mieux, FOR I = 1 TO...:NEXT ; en comptant 1000 par seconde, environ... Certains programmes exigent des temps ultra-précis, en ce cas utilisez ce ''module 9000''.

```
800     EXEMPLE DE DELAI PRECIS
810 FOR I=1 TO 2000:NEXT:PRINT CHR$(7)
820 DUR=5.78:GOSUB 9000
830 PRINT CHR$(7)
840 END
9000  '  PAUSE PRECISE EN SECONDES
9010 TF=TIME+DUR*300
9020 WHILE TIME<TF:WEND
9030 RETURN
```

LE SUPER JOY-STICK

A lire certains bancs d'essai de la presse et le manuel, vous en avez déduit que le JOY-STICK n'avait que quatre directions. Cela nous semblant anormal, nous avons décortiqué l'image binaire du nombre renvoyé par la fonction JOY. Il y a bel et bien *huit* directions :

Haut	00001 = 1
Bas	00010 = 2
Gauche	00100 = 4
Haut-gauche	00101 = 5
Bas-gauche	00110 = 6
Droite	01000 = 8
Bas-droite	01010 = 9
Haut-droite	01100 = 12
Fire 1	10000 = 16
Fire 2	100000 = 32

Peu importe la marque du JOY-STOCK dont vous disposez car les prises sont normalisées (c'est bien l'unique chose qui ne varie pas d'un micro-ordinateur à un autre...).

Nous vous proposons deux programmes de démonstration. Le premier permet de gribouiller sur l'écran, par PLOT, c'est donc assez lent. Le second affiche la lettre X par PRINT, c'est donc rapide.

Dans les deux programmes, l'appui simultané sur la touche FIRE permet de se déplacer sans tracer, et d'effacer un trait existant avec clignotement du repère.

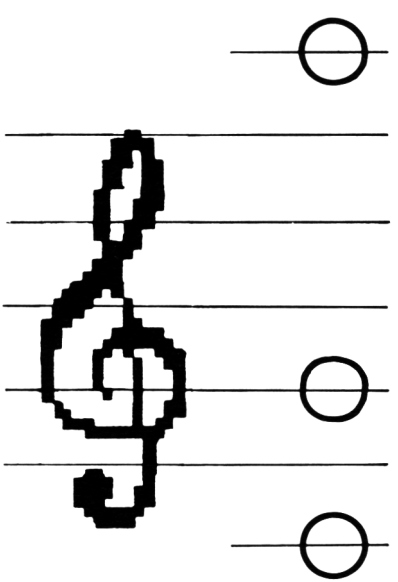
```
10 ' DESSIN AVEC JOY-STICK : FIRE=PAS DE
 TRACE
20 CLS:BORDER 9:DEFINT H,V,T
30 H=320:V=200
40 B$=BIN$(JOY(0),5)
50 H=H+VAL(MID$(B$,2,1))-VAL(MID$(B$,3,1
))
60 V=V+VAL(MID$(B$,5,1))-VAL(MID$(B$,4,1
))
70 T=VAL(LEFT$(B$,1))
80 PLOT H,V,1:IF T=1 THEN PLOT H,V,0
90 GOTO 40
99 '-----------------------------------------
100 'CURSEUR AVEC JOY-STICK: FIRE=PAS DE
 TRACE
110 MODE 1:BORDER 9:DEFINT H,V,T
120 B$=BIN$(JOY(0),5)
130 H=H+VAL(MID$(B$,2,1))-VAL(MID$(B$,3,
1))
140 V=V-VAL(MID$(B$,5,1))+VAL(MID$(B$,4,
1))
150 T=VAL(LEFT$(B$,1))
160 IF H>40 THEN H=40
```

```
170 IF H<1 THEN H=1
180 IF V>25 THEN V=25
190 IF V<1 THEN V=1
200 LOCATE H,V:PRINT "x"
210 IF T=1 THEN LOCATE H,V:PRINT " "
220 GOTO 120
230 END
290 '  FIN DE LISTING
```

BRUIT OU MUSIQUE ?

Programmer un air musical est beaucoup plus facile que de créer un bruit intéressant.

Pour reproduire un air, il faut la partition musicale. Vous ne savez pas lire les notes sur une ''portée'' ? On a prévu cela, et la figure XVI 1 vous suffira.



34	LA	22	
32	SOL	20	
30	FA	18	
29	MI	17	
27	RE	15	
25	DO	13	
24	SI	12	36
22	LA	10	34
20	SOL	8	32
18	FA	6	30
17	MI	5	29
15	RE	3	27
13	DO	1	25

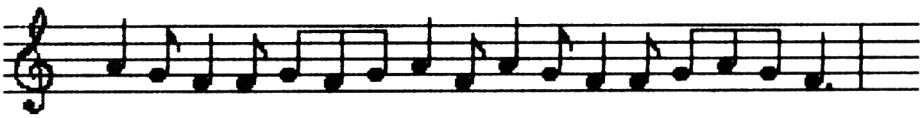
(0 = silence)

Figure XVI 1
Chaque note est remplacée par un numéro, d'où écriture rapide d'une mélodie.

Traduire la note lue en ''période AMSTRAD'' est une chose très fastidieuse, voire rebutante. Un sous-programme (ligne 57000) traduit les périodes des trois octaves les plus utilisées en numéros 0 à 36 ; zéro = silence, 1 est la note la plus grave, et 36 la plus aiguë. La progression se fait par demi-tons. Ainsi, FA # (dièse) = n° 19 ou 7 ou 31 selon l'octave ; de même, SI bémol = 23. Le LA du diapason (440 Hertz) correspond au n° 22.

La figure XVI 2 vous indique la durée à donner aux différents types de notes. Pour la ''noire'', nous l'avons fixée à 60 (un musicologue nous l'a conseillé).

Fais Dodo



𝅗𝅥	blanche	120
♩	noire	60
♪	croche	30
𝅘𝅥𝅯	double croche	15
♩.	noire pointée	60 x 1,5 = 90

Figure XVI.2 : La partition que nous avons transcrite en BASIC

La première note de cette mélodie est donc SOUND 1, NOTE(22), 60,15 (1 = canal ; 15 = volume maxi), mais en boucle FOR NEXT, c'est vite écrit (ligne 290).

```
10     "FAIS DODO" ECRITURE NORMALE
20 DATA 284,60,319,30,358,60,3822,5
30 DATA 358,30,319,30,358,30,319,30
40 DATA 284,60,358,30,284,60,319,30
50 DATA 358,60,3822,5,358,30,319,30
60 DATA 284,30,319,30,358,90
70 FOR I=1 TO 19
80 READ P,D
90 SOUND 1,P,D,15
100 NEXT
110 END
199 '-----------------------------------------------
200 ' "FAIS DODO" AVEC NOTES CODEES
210 GOSUB 57000
220 DATA 22,60,20,30,18,60,0,5,18,30
230 DATA 20,30,18,30,20,30,22,60
240 DATA 18,30,22,60,20,30,18,60,0,5
250 DATA 18,30,20,30,22,30,20,30
260 DATA 18,90
270 RESTORE 220
280 FOR I=1 TO 19:READ N,D
290 SOUND 1,NOTE(N),D,15:NEXT
300 END
57000 ' NOTES DE LA GAMME
57010 DATA 3822
57020 DATA 956,902,851,804,758,716,676,6
38,602,568,536,506
57030 DATA 478,451,426,402,379,358,338,3
19,301,284,268,253
57040 DATA 239,225,213,201,190,179,169,1
59,150,142,134,127
57050 DIM NOTE(36):RESTORE 57010
57060 FOR I=0 TO 36:READ NOTE(I):NEXT
57070 RETURN
```

Le module en 57000 sera sauvegardé sur la cassette UTIL. Il établit les 36 périodes en DIM, l'indice zéro (silence) correspond à un son tellement grave (30 Hz) que bien peu d'enceintes peuvent reproduire.

Nous avons écrit deux versions du même air ; de 10 à 110 l'écriture ''classique'' où les DATA sont très pénibles à taper, et notre version (RUN 200) dans laquelle les lignes de DATA sont environ trois à quatre fois plus rapides à écrire, et aussi plus faciles à modifier.

Ainsi, pour ''baisser'' d'un demi-ton, on diminue le numéro d'une unité.

Il y a 17 notes + 2 silences. Quel rôle ont ces silences ? Pour séparer *deux notes identiques* comme la troisième et la quatrième, une durée très courte (5) suffit pour simuler l'attaque de la suivante.

Le résultat musical ne vous satisfait qu'à moitié ? Seule la ligne 290 sera à modifier : avec SOUND 1, NOTE(N + 12), D*1,2,15 vous jouerez un octave plus haut et un peu plus lentement. Vous pouvez aussi faire NOTE(N + 6).

S'il y a plusieurs airs dans votre programme, ne refaites pas GOSUB 57000 une seconde fois !

Pour les bruits, il vous faudra combiner les fonctions SOUND, ENV et ENT avec beaucoup de patience et de flair, car les bruits avec SOUND seuls sont plutôt rares, à part :

```
SOUND 1,0,40,15
```

qui fait un ''TOC, TOC''...

Aller plus loin dans les fonctions sonores serait de l'acoustique plus de la programmation en Basic, et sortirait ainsi du cadre de ce livre.

Chapitre XVII

MAGNÉTOPHONE ET IMPRIMANTE

LE MAGNETOPHONE

Beaucoup d'acheteurs français ignorent que la marque AMSTRAD existe depuis longtemps comme fabricant anglais de matériel HI-FI ; le CPC 464 est le premier de son nouveau département informatique. On devine alors que le magnéto-cassette incorporé a dû être ''prélevé'' dans sa gamme habituelle, donc un banal (et standard) magnétophone monophonique. Le seul reproche concerne l'absence d'arrêt automatique en fin de bobinage. L'oubli prolongé, après blocage des touches ''REW'' ou ''FF'', pourrait provoquer une usure prématurée des courroies ou des freins de bobines. D'autre part, nous manipulons ses touches avec douceur car elles pourraient ne pas avoir la robustesse mécanique des touches du clavier.

Le manuel d'origine pêche par excès de prudence en préconisant la vitesse lente de 1000 bauds ($\simeq$ 100 octets/seconde). Nous avons toujours enregistré à 2000 bauds par :

SPEED WRITE 1:SAVE''

sans jamais un seul raté, même avec des cassettes de qualité médiocre.

Si un jour vous aviez un ennui de lecture, il suffira de nettoyer la tête magnétique avec UNIQUEMENT un coton-tige imbibé d'alcool ; il faut parfois frotter pendant cinq secondes. Pas d'autre solvant que l'alcool !

QUELLES CASSETTES UTILISER ?

Surtout pas des cassettes à l'oxyde de chrome car, une fois enregistrées, ce magnétophone ne pourrait les effacer complètement...

Evitez les C90 et prohibez les C120. Par économie, nous utilisons des C60, normales, mais de bonnes marques, parce qu'étant meilleur marché que les C15 et C30 de marques inconnues. La loi des grandes et petites séries.

Les principaux ennemis des bandes et des magnétophones sont poussière et traces de doigts ; donc ces recommandations :

- Le couvercle du magnéto ne doit pas rester ouvert,
- toujours rembobiner une cassette, car des traces de doigts sur la bande-amorce ont peu de conséquences,
- ne laissez jamais traîner sur la table une cassette hors de son boîtier, surtout si vous êtes fumeur (cendres).

DE LA METHODE ET QUELQUES TRUCS

Par le bruit du haut-parleur, vous avez sans doute remarqué que l'ordinateur prévoit un "blanc" avant et après un enregistrement. C'est une double précaution ; toutefois, le blanc du départ dure moins longtemps que le passage de la bande amorce ! D'où le mode opératoire suivant en début de cassette.

- Compteur à zéro ;
- "FF" jusqu'à "1" ou "2" au compteur. Là vous pouvez enregistrer ;
- si vous avez plusieurs enregistrement à faire sur une même cassette, partez sur des multiples de 10 au compteur, mais en laissant 2 à 3 "points de sécurité" : si un enregistrement se termine en "18" ou "19", démarrez le prochain non pas en "20", mais en "30" (celui terminant en "18" peut être allongé par la suite...) ;
- notez les repères compteurs sur l'étiquette du coffret plastique ;
- pour ajuster le bobinage sur un repère précis (à ± un demi point), on a davantage de précision en "FF" qu'en "REW" ;
- lorsque vous mettez au point un programme, faites *alternativement* les sauvegardes (avec date et heure) sur *deux cassettes différentes,* chacune en début de bobine. En voici la raison : si vous "écrasez" chaque fois l'ancienne version par la nouvelle, que se passerait-il si une coupure de courant intervenait en cours de sauvegarde ? Vous perdriez TOUT... ;
- si votre programme est important, enregistrez-le deux fois à la file, cassez la languette de protection et ne mettez aucun autre programme dessus, même s'il s'agit d'une C60 ;
- prévoyez une C60 de sauvegarde-archivage de sécurité où vous mettrez tous vos programmes, mais avec de larges blancs entre eux au cas où une remise à jour de l'un d'eux déborde largement sur l'ancienne version. Rappelez-vous qu'un programme devient non rechargeable s'il manque un peu du début ou un peu de la fin ;

— vous pouvez dupliquer rapidement une de vos cassettes en effectuant une ''copie physique'' entre deux autres magnétophones, de préférence HI-FI : pas de DOLBY et contrôle manuel du volume (pas de C.A.G. = Contrôle automatique de gain). Attention, les nouvelles lois contre le piratage de logiciels sont devenues très très sévères. Pensez à protéger certains de vos programmes que vous prêtez à des ''amis'' : passez leur une version que vous aurez enregistrée par SAVE'' '',P ;
— enfin le grand danger : cette touche ''PAUSE'' qui non seulement est inutile mais peut générer bien des bêtises. Condamnez-la avec un petit carré de carton fixé par de l'adhésif.

C'est souvent qu'on l'enfonce par erreur, et si vous faites un SAVE, il n'y aura rien pour vous prévenir que vous enregistrez dans le vide. Vérifiez toujours que les bobines tournent.

UTILISATION D'UNE IMPRIMANTE

Il existe deux techniques de liaison entre un micro-ordinateur et une imprimante :
— Le standard ''CENTRONICS'' ou ''Parallèle'' est de loin le plus commode et le plus répandu. C'est celui qu'utilise l'AMSTRAD.
— Le standard (?) ''RS 232C'' ou ''Série'', assez rare, sauf sur les micro-ordinateurs portatifs.

Il y a trois technologies d'imprimante :
1) A aiguilles ou ''matricielles''. Elles sont très rapides (80 à 160 caractères/seconde), très robustes, très souples d'emploi (nombreuses formes de caractères, options programmées, etc.). Ce sont les bêtes à tout faire. Deux défauts : elles sont chères, elles sont bruyantes.
2) A stylos-bille. Elles savent aussi dessiner et en quatre couleurs. Leurs options d'écriture sont limitées. Elles sont excessivement lentes. Le tracé est net ; les recharges de stylos spéciaux peuvent devenir une rente...
3) Thermiques. Elles utilisent un papier spécial qui bleuit à la chaleur. Bon marché, assez rapies, ultra silencieuses. Hélas, leur tracé est souvent mal lisible.

Un imprimante est un ''investissement à vie'' : on change de micro-ordinateur, mais pas d'imprimante, même dans le milieu professionnel. Voilà pourquoi on n'en trouve pratiquement jamais d'occasion, surtout en matricielles.

Avant de fixer votre choix sur un modèle, faites-lui imprimer des ''y'' et des ''g'' minuscules afin de vérifier que le jambage inférieur descend bien au-dessous de l'alignement normal. Le contraire serait gênant en ''traitement de texte'', car il n'y aurait pas la ''qualité courrier''.

LE MEMOIRE D'UNE IMPRIMANTE

L'électronique d'une imprimante rappelle beaucoup celle d'un micro-ordinateur. On a une mémoire morte résidente ROM et une mémoire vive RAM que l'on appelle communément le ''buffer''.

Le mode d'impression, c'est-à-dire la forme des caractères, leur taille, etc., peut être fixé de deux manières : par de minuscules interrupteurs situés sous le capot (= SWITCHES), ou par des ordres Basic venant de l'ordinateur. Exemple :

```
PRINT #8,CHR$(27);''4''; (pour EPSON)
```

ordonne d'imprimer ce qui suit en caractères italiques. Cette ''consigne'' est mise dans le buffer et elle *reste active* tant qu'il n'y a pas de contr'ordre, ou extinction de l'imprimante. Il en est de même pour la grande majorité des ordres. Ces ordres en Basic sont prioritaires sur les positions des SWITCHES.

Pour fixer les idées, la classique EPSON RX 80 dispose de 53 commandes différentes, et dont beaucoup sont paramétrables. Une sorte de second Basic ? Hélas non, car ces syntaxes diffèrent souvent d'une marque à une autre.

L'une de ces commandes s'appelle ''Initialisation'', c'est un RESET qui vide le buffer ; on se retrouve alors avec les options par défaut (établies par les SWITCHES) que l'on a lors de la mise sous tension. Une bonne habitude est de glisser cette commande au début et à la fin d'un programme d'édition sur imprimante. Cela évite bien des mauvaises surprises...

Fait très rare, l'AMSTRAD possède une fonction Basic destinée à *toutes* les imprimantes, c'est WIDTH qui fixe le nombre maximal de caractères par ligne ; par exemple WIDTH 40 va envoyer un ''retour chariot'', un CHR$(13), après le quarantième caractère. Cette consigne reste active dans le ''buffer ordinateur'' même après NEW ou une extinction et remise sous tension de l'imprimante.

On peut obtenir le même résultat en envoyant un certain ordre à l'imprimante, exemple, pour une EPSON :

```
PRINT #8,CHR$(27)''Q''CHR$(40)
```

mais il serait dans le buffer de l'imprimante et non dans celui de l'ordinateur. Au choix !

LES ORDRES COURANTS (pour toutes imprimantes)

LIST #8 : fait le listing d'un programme.
LIST 100-400,#8 : listing des lignes 100 à 400.
LIST 100-,#8 : listing à partir de la ligne 100.
LIST -400,#8 : listing jusqu'à la ligne 400.

Evitez d'utiliser les fonctions de tabulations propres à l'imprimante, elles sont complexes et réservent souvent de mauvaises surprises. Utilisez plutôt le Basic de l'ordinateur.

PRINT #8,NB,NC : décalage de 13 (=zone).
PRINT #8,USING "# # #.#";NB (=idem que sur écran).
PRINT #8,SPC(10);"ICI" : à partir du 11e caractère.
PRINT #8,TAB(11);"ICI" : même effet.
PRINT #8,SPACE$(10);"ICI" : même effet.

Pour sauter quatre lignes :

```
FOR I=1 TO 4:PRINT #8:NEXT
```

En combinant cette ligne avec la fonction TAB illustrée plus haut, vous disposez d'une sorte de LOCATE relatif.

LES "CLAVIERS" INTERNATIONAUX

On appelle "claviers" le jeu de caractères ASCII propre à chaque pays. Les différences concernent certains caractères spéciaux propres à une langue. Pour la France, il s'agit surtout des voyelles accentuées (voir notre sous-programme AZERTY du chapitre IX).

Les bonnes imprimantes à aiguilles possèdent plusieurs "claviers" exécutables par des SWITCHES et par ordre venant de l'ordinateur. L'AMSTRAD étant en QWERTY anglo-saxon (c'est bien mieux pour programmer que l'AZERTY), nous vous conseillons de "SWITCHER" votre imprimante en clavier USA, les listings reproduiront bien ce qu'il y a sur les touches ou à l'écran.

Si vous avez utilisé le sous-programme AZERTY pour avoir les caractères français à l'écran, et si ce programme a une partie édition sur imprimante, envoyez à celle-ci l'ordre d'imprimer en caractères français, et n'oubliez pas d'annuler cet ordre en fin de programme.

NOTE : Ne pas confondre les mots "EDITEUR" et "EDITION". Le premier concerne la modification des lignes BASIC d'un programme, le second signifie écriture sur imprimante.

Notre dernier conseil relatif aux imprimantes va sans doute vous surprendre, mais il est dicté par de nombreuses expériences vécues. Prenez le temps de bien lire TOUTE la notice..., même si elle est en anglais, ce qui arrive souvent. Le but est pour vous de savoir *que telle option existe,* car il est stupide, mais malheureusement courant, qu'un programmeur se creuse la tête pour résoudre un problème d'édition en vingt lignes de Basic, alors que la ROM de son imprimante possède une fonction qui fait précisément cette chose-là !

LES PHOTOS D'ECRANS

Si vous voulez ''vendre'' un programme original à une revue, il est souvent souhaitable d'y joindre quelques photos d'écran de certains passages. Cela demande peu de matériel mais de bonnes méthodes car c'est moins facile que l'on pourrait croire.

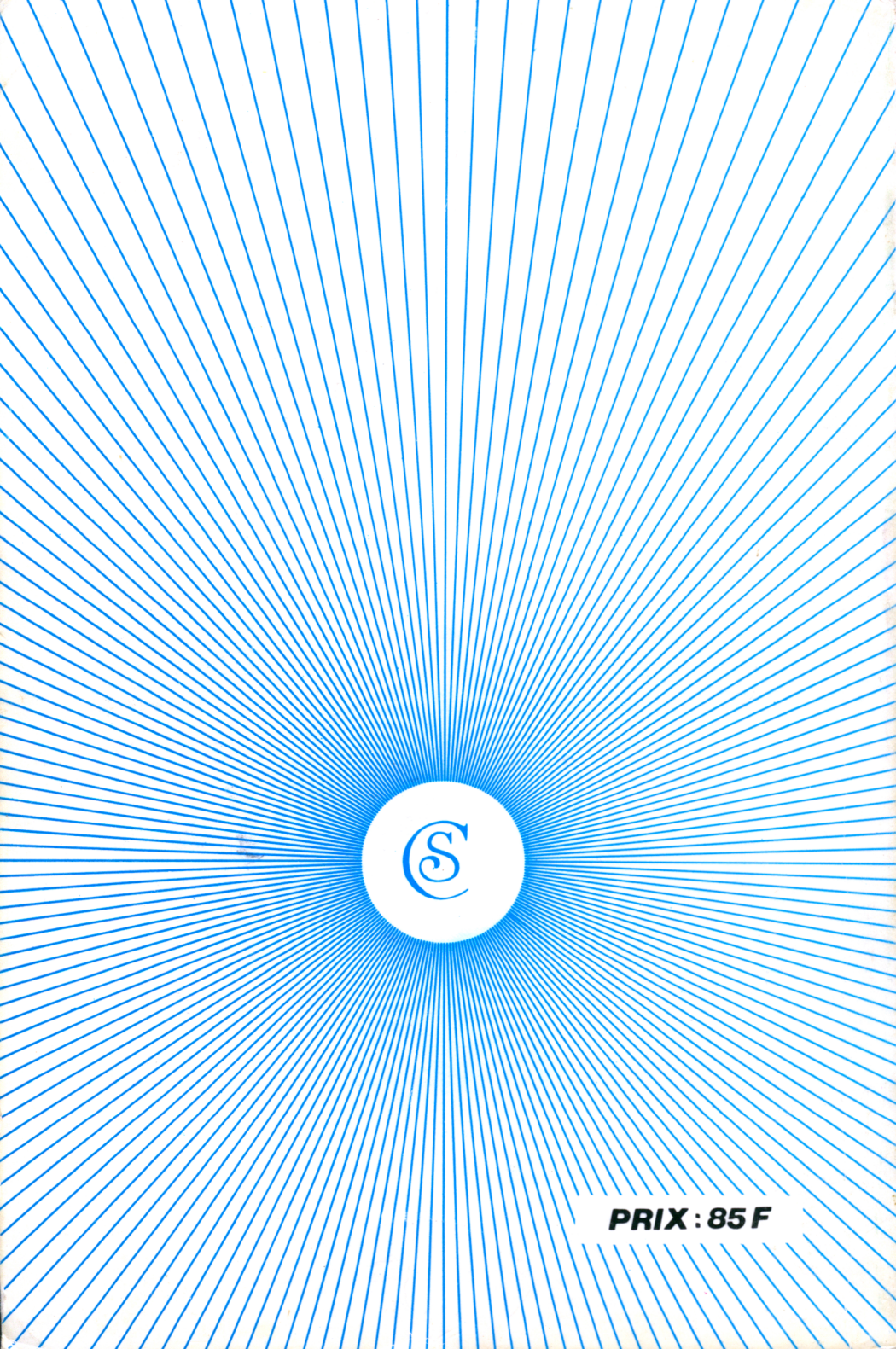
1) Nettoyez l'écran, pas seulement avec un chiffon, mais avec du coton imbibé d'alcool et de la patience. En effet, l'électricité statique a accumulé une couche de crasse très tenace. Elle ne vous choque pas, mais en photo, on ne voit qu'elle.
2) Votre appareil photo est sûrement équipé d'un obturateur à rideaux, alors la vitesse la plus rapide sera 1/4 seconde (1/15 seconde pour un obturateur central). Avec des vitesses plus rapides, vous aurez droit à des zones sombres, des barres obliques (ou horizontales avec un obturateur central) ; c'est un phénomène stroboscopique. Votre œil ne le voit pas, mais l'émulsion photographique n'a pas de ''persistance rétinienne''...
3) L'appareil est sur trépied, bien en face de l'écran (ce n'est pas facile). Méfiez-vous des zooms qui apportent souvent des déformations géométriques dans les angles aux courtes distance de mise au point. Avec du 100 ASA (sensibilité maximum), film couleur du jour pour des diapos, réglez l'appareil sur 1/4 s et f:11. Ecran en luminosité normale.
4) Faites l'obscurité dans la pièce. Sinon vous aurez le reflet flou de votre appareil sur le verre de l'écran (c'est grotesque). Faites plusieurs clichés avec des ouvertures différentes (+ 1 et − 1 cran de diaphragme autour de la valeur indiquée plus haut, ou celle de la cellule).

Comme vous le constatez, ce n'est pas une question de matériel sophistiqué, mais seulement quelques menus détails pratiques qui font toute la différence.

SOMMAIRE

Chapitre I	Les deux urgences	9
Chapitre II	La pratique de l'éditeur	17
Chapitre III	La programmation dite structurée	23
Chapitre IV	L'organigramme	29
Chapitre V	Numéros de lignes et variables	35
Chapitre VI	Les ressources des fonctions chaîne	41
Chapitre VII	Manipulons les fonctions numériques	49
Chapitre VIII	Les couleurs	61
Chapitre IX	Les belles pages d'écran	69
Chapitre X	Entrez sans planter	79
Chapitre XI	Les boucles et leurs pièges	87
Chapitre XII	La gestion des données DIM, DATA, FICHIER	93
Chapitre XIII	La chasse aux bugs	101
Chapitre XIV	Le graphisme simple	107
Chapitre XV	La mémoire d'écran	117
Chapitre XVI	Temps, joy et musique	127
Chapitre XVII	Magnétophone et imprimante	137

Impression Jouve - Mayenne
N° éditeur 46. Composition Fideltex
N° d'impression : 14992. Dépôt légal : Novembre 1985



PRIX : 85 F

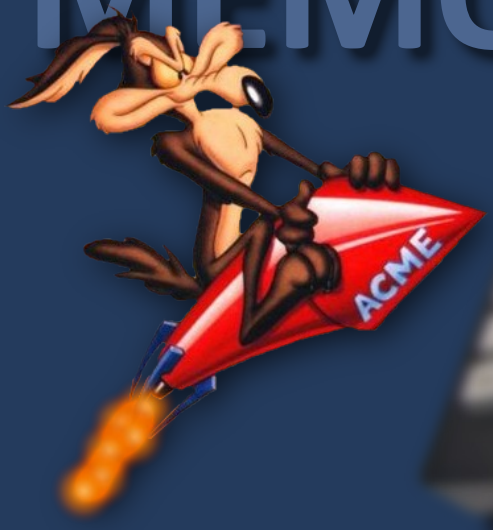
MIEUX PROGRAMMER SUR AMSTRAD

M. ARCHAMBAULT

Document numérisé avec amour par :

AMSTRAD CPC

MÉMOIRE ÉCRITE



https://acpc.me/